First-order logic with game-theoretic recursion

Reijo Jaakkola
reijo.jaakkola@tuni.fi

Tampere University

April 20, 2023

Computational logic CL

- CL was introduced by (Kuusisto, 14), where it was also proved that it characterises the class of recursively enumerable languages.

Computational logic CL

- CL was introduced by (Kuusisto, 14), where it was also proved that it characterises the class of recursively enumerable languages. More precisely, for every Turing machine M we can find a sentence ϕ_M of CL s.t. for every finite (relational) structure $\mathfrak{A}$ the following conditions hold.

Computational logic CL

- CL was introduced by (Kuusisto, 14), where it was also proved that it characterises the class of recursively enumerable languages. More precisely, for every Turing machine M we can find a sentence ϕ_M of CL s.t. for every finite (relational) structure $\mathfrak{A}$ the following conditions hold.

  1. $\mathfrak{A} \models \phi_M$ iff M accepts $\mathfrak{A}$

Computational logic CL

- CL was introduced by (Kuusisto, 14), where it was also proved that it characterises the class of recursively enumerable languages. More precisely, for every Turing machine M we can find a sentence ϕ_M of CL s.t. for every finite (relational) structure $\mathfrak{A}$ the following conditions hold.

  1. $\mathfrak{A} \models \phi_M$ iff M accepts $\mathfrak{A}$
  2. $\mathfrak{A} \models \neg\phi_M$ iff M rejects $\mathfrak{A}$

Computational logic CL

- CL was introduced by (Kuusisto, 14), where it was also proved that it characterises the class of recursively enumerable languages. More precisely, for every Turing machine M we can find a sentence ϕ_M of CL s.t. for every finite (relational) structure $\mathfrak{A}$ the following conditions hold.
  1. $\mathfrak{A} \models \phi_M$ iff M accepts $\mathfrak{A}$
  2. $\mathfrak{A} \models \neg\phi_M$ iff M rejects $\mathfrak{A}$
  3. ϕ_M is indeterminate on $\mathfrak{A}$ iff the computation of M on $\mathfrak{A}$ diverges

Computational logic CL

- CL was introduced by (Kuusisto, 14), where it was also proved that it characterises the class of recursively enumerable languages. More precisely, for every Turing machine M we can find a sentence ϕ_M of CL s.t. for every finite (relational) structure $\mathfrak{A}$ the following conditions hold.

  1. $\mathfrak{A} \models \phi_M$ iff M accepts $\mathfrak{A}$
  2. $\mathfrak{A} \models \neg\phi_M$ iff M rejects $\mathfrak{A}$
  3. ϕ_M is indeterminate on $\mathfrak{A}$ iff the computation of M on $\mathfrak{A}$ diverges

  Conversely, if ϕ is a sentence of CL, then $\mathrm{Mod}(\phi)$ is a recursively enumerable set of models.

Computational logic CL

- CL was introduced by (Kuusisto, 14), where it was also proved that it characterises the class of recursively enumerable languages. More precisely, for every Turing machine M we can find a sentence ϕ_M of CL s.t. for every finite (relational) structure $\mathfrak{A}$ the following conditions hold.

  1. $\mathfrak{A} \models \phi_M$ iff M accepts $\mathfrak{A}$
  2. $\mathfrak{A} \models \neg\phi_M$ iff M rejects $\mathfrak{A}$
  3. ϕ_M is indeterminate on $\mathfrak{A}$ iff the computation of M on $\mathfrak{A}$ diverges

  Conversely, if ϕ is a sentence of CL, then $\mathrm{Mod}(\phi)$ is a recursively enumerable set of models.

- CL extends standard first-order logic FO with two natural features.

Computational logic CL

- CL was introduced by (Kuusisto, 14), where it was also proved that it characterises the class of recursively enumerable languages. More precisely, for every Turing machine M we can find a sentence ϕ_M of CL s.t. for every finite (relational) structure $\mathfrak{A}$ the following conditions hold.
  1. $\mathfrak{A} \models \phi_M$ iff M accepts $\mathfrak{A}$
  2. $\mathfrak{A} \models \neg\phi_M$ iff M rejects $\mathfrak{A}$
  3. ϕ_M is indeterminate on $\mathfrak{A}$ iff the computation of M on $\mathfrak{A}$ diverges

  Conversely, if ϕ is a sentence of CL, then $\mathrm{Mod}(\phi)$ is a recursively enumerable set of models.
- CL extends standard first-order logic FO with two natural features.
  1. The ability to modify the underlying model: adding new elements to the domain of the model, new tuples to relations, new relations etc.

Computational logic CL

- CL was introduced by (Kuusisto, 14), where it was also proved that it characterises the class of recursively enumerable languages. More precisely, for every Turing machine M we can find a sentence ϕ_M of CL s.t. for every finite (relational) structure $\mathfrak{A}$ the following conditions hold.

  1. $\mathfrak{A} \models \phi_M$ iff M accepts $\mathfrak{A}$
  2. $\mathfrak{A} \models \neg\phi_M$ iff M rejects $\mathfrak{A}$
  3. ϕ_M is indeterminate on $\mathfrak{A}$ iff the computation of M on $\mathfrak{A}$ diverges

  Conversely, if ϕ is a sentence of CL, then $\mathrm{Mod}(\phi)$ is a recursively enumerable set of models.

- CL extends standard first-order logic FO with two natural features.

  1. The ability to modify the underlying model: adding new elements to the domain of the model, new tuples to relations, new relations etc.
  2. The ability to use recursion (looping) via self-reference.

Recursion via self-reference

- Logics with different kinds of recursive looping capacities have been widely studied in the context of finite model theory.

Recursion via self-reference

- Logics with different kinds of recursive looping capacities have been widely studied in the context of finite model theory.

  1. $\mathrm{FO}[\mathrm{TC}]$: extension of FO that can compute the transitive closure of a binary relation.

$$\forall x \forall y(\neg x = y \rightarrow [\mathrm{TC}_{x,y} Exy]xy)$$

Recursion via self-reference

- Logics with different kinds of recursive looping capacities have been widely studied in the context of finite model theory.

    1. $\mathrm{FO}[\mathrm{TC}]$: extension of FO that can compute the transitive closure of a binary relation.

    $$\forall x \forall y(\neg x = y \rightarrow [\mathrm{TC}_{x,y} Exy]xy)$$

    Captures NL over ordered finite structures.

Recursion via self-reference

- Logics with different kinds of recursive looping capacities have been widely studied in the context of finite model theory.

1. $\mathrm{FO[TC]}$: extension of FO that can compute the transitive closure of a binary relation.

$$\forall x \forall y(\neg x = y \rightarrow [\mathrm{TC}_{x,y} Exy]xy)$$

   Captures NL over ordered finite structures.

2. LFP: extension of FO with the least fixed point operator.

$$\forall x \forall y(\neg x = y \rightarrow [\mathrm{LFP}_{x,y,X}(Exy \vee \exists z(Xxz \wedge Ezy))]xy)$$

Recursion via self-reference

- Logics with different kinds of recursive looping capacities have been widely studied in the context of finite model theory.

  1. $\mathrm{FO}[\mathrm{TC}]$: extension of FO that can compute the transitive closure of a binary relation.

  $$\forall x \forall y(\neg x = y \to [\mathrm{TC}_{x,y} Exy]xy)$$

  Captures NL over ordered finite structures.

  2. LFP: extension of FO with the least fixed point operator.

  $$\forall x \forall y(\neg x = y \to [\mathrm{LFP}_{x,y,X}(Exy \vee \exists z(Xxz \wedge Ezy))]xy)$$

  Captures P over ordered finite structures.

Recursion via self-reference

- Logics with different kinds of recursive looping capacities have been widely studied in the context of finite model theory.

  1. $\mathrm{FO[TC]}$: extension of FO that can compute the transitive closure of a binary relation.

  $$\forall x \forall y(\neg x = y \rightarrow [\mathrm{TC}_{x,y} Exy]xy)$$

  Captures NL over ordered finite structures.

  2. LFP: extension of FO with the least fixed point operator.

  $$\forall x \forall y(\neg x = y \rightarrow [\mathrm{LFP}_{x,y,X}(Exy \vee \exists z(Xxz \wedge Ezy))]xy)$$

  Captures P over ordered finite structures.

- In CL the recursion is implemented via self-reference (or go-to statements).

$$\forall x \forall y(\neg x = y \rightarrow L(Exy \vee \exists z(Exz \ \wedge \ \exists x(x = z \wedge C_L))))$$

Recursion via self-reference

- Logics with different kinds of recursive looping capacities have been widely studied in the context of finite model theory.

  1. $\mathrm{FO[TC]}$: extension of FO that can compute the transitive closure of a binary relation.

     $$\forall x \forall y(\neg x = y \rightarrow [\mathrm{TC}_{x,y} Exy]xy)$$

     Captures NL over ordered finite structures.

  2. LFP: extension of FO with the least fixed point operator.

     $$\forall x \forall y(\neg x = y \rightarrow [\mathrm{LFP}_{x,y,X}(Exy \vee \exists z(Xxz \wedge Ezy))]xy)$$

     Captures P over ordered finite structures.

- In CL the recursion is implemented via self-reference (or go-to statements).

$$\forall x \forall y(\neg x = y \rightarrow L(Exy \vee \exists z(Exz \ \wedge \ \exists x(x = z \wedge C_L))))$$

Formulas are interpretated using *game-theoretical* semantics.

Recursion via self-reference

- Logics with different kinds of recursive looping capacities have been widely studied in the context of finite model theory.
  1. $\mathrm{FO[TC]}$: extension of FO that can compute the transitive closure of a binary relation.

     $$\forall x \forall y(\neg x = y \to [\mathrm{TC}_{x,y} Exy]xy)$$

     Captures NL over ordered finite structures.
  2. LFP: extension of FO with the least fixed point operator.

     $$\forall x \forall y(\neg x = y \to [\mathrm{LFP}_{x,y,X}(Exy \vee \exists z(Xxz \wedge Ezy))]xy)$$

     Captures P over ordered finite structures.
- In CL the recursion is implemented via self-reference (or go-to statements).

  $$\forall x \forall y(\neg x = y \to L(Exy \vee \exists z(Exz \ \wedge \ \exists x(x = z \wedge C_L))))$$

  Formulas are interpretated using *game-theoretical* semantics. We call the extension of FO with this type of recursion SCL (static computational logic).

Overview of the rest of the talk

1. Syntax & Semantics of SCL.
2. Validity problem of SCL.
3. Some results on the model theory of SCL.

Syntax of SCL

Definition

Fix a countable set $\mathrm{LBS} = \{L_n \mid n \in \mathbb{N}\}$ of *label* symbols.

Syntax of SCL

Definition

Fix a countable set $\mathrm{LBS} = \{L_n \mid n \in \mathbb{N}\}$ of *label* symbols. For reach relational vocabulary τ the set of formulas $\mathrm{SCL}[\tau]$ is defined by the following grammar:

$$\phi ::= x = y \mid R(\overline{x}) \mid C_L \mid \neg\phi \mid \phi \wedge \phi \mid \exists x \phi \mid L\phi,$$

where $R \in \tau$ and $L \in LBS$.

Game-theoretical semantics (GTS) for SCL

- We associate to each τ-structure $\mathfrak{A}$, assignment s and a formula ϕ of $\mathrm{SCL}[\tau]$ a two-player game $\mathcal{G}_\infty(\mathfrak{A}, s, \phi)$, which is played by Verifier and Falsifier.

Game-theoretical semantics (GTS) for SCL

- We associate to each τ-structure $\mathfrak{A}$, assignment s and a formula ϕ of $\mathrm{SCL}[\tau]$ a two-player game $\mathcal{G}_\infty(\mathfrak{A}, s, \phi)$, which is played by Verifier and Falsifier. We then define that

$$\mathfrak{A}, s \models \phi \Leftrightarrow \text{ Verifier has a winning strategy in the game } \mathcal{G}_\infty(\mathfrak{A}, s, \phi).$$

Game-theoretical semantics (GTS) for SCL

- We associate to each τ-structure $\mathfrak{A}$, assignment s and a formula ϕ of $\mathrm{SCL}[\tau]$ a two-player game $\mathcal{G}_\infty(\mathfrak{A}, s, \phi)$, which is played by Verifier and Falsifier. We then define that

$$\mathfrak{A}, s \models \phi \Leftrightarrow \text{ Verifier has a winning strategy in the game } \mathcal{G}_\infty(\mathfrak{A}, s, \phi).$$

  We call $\mathcal{G}_\infty$ the **unbounded evaluation game**.

- Positions of the game are triples $(r, \psi, \#)$, where r is the current assignment, ψ is a subformula of ϕ and $\# \in \{-, +\}$.

Game-theoretical semantics (GTS) for SCL

- We associate to each τ-structure $\mathfrak{A}$, assignment s and a formula ϕ of $\mathrm{SCL}[\tau]$ a two-player game $\mathcal{G}_\infty(\mathfrak{A}, s, \phi)$, which is played by Verifier and Falsifier. We then define that

$$\mathfrak{A}, s \models \phi \Leftrightarrow \text{ Verifier has a winning strategy in the game } \mathcal{G}_\infty(\mathfrak{A}, s, \phi).$$

  We call $\mathcal{G}_\infty$ the **unbounded evaluation game**.

- Positions of the game are triples $(r, \psi, \#)$, where r is the current assignment, ψ is a subformula of ϕ and $\# \in \{-, +\}$. Initial position is just $(s, \phi, +)$.

Game-theoretical semantics (GTS) for SCL

- We associate to each τ-structure $\mathfrak{A}$, assignment s and a formula ϕ of $\mathrm{SCL}[\tau]$ a two-player game $\mathcal{G}_\infty(\mathfrak{A}, s, \phi)$, which is played by Verifier and Falsifier. We then define that

$$\mathfrak{A}, s \models \phi \Leftrightarrow \text{ Verifier has a winning strategy in the game } \mathcal{G}_\infty(\mathfrak{A}, s, \phi).$$

  We call $\mathcal{G}_\infty$ the **unbounded evaluation game**.

- Positions of the game are triples $(r, \psi, \#)$, where r is the current assignment, ψ is a subformula of ϕ and $\# \in \{-, +\}$. Initial position is just $(s, \phi, +)$.
- Rules for first-order connectives and atomic formulas are standard. Some examples.

Game-theoretical semantics (GTS) for SCL

- We associate to each τ-structure $\mathfrak{A}$, assignment s and a formula ϕ of $\mathrm{SCL}[\tau]$ a two-player game $\mathcal{G}_\infty(\mathfrak{A}, s, \phi)$, which is played by Verifier and Falsifier. We then define that

  $$\mathfrak{A}, s \models \phi \Leftrightarrow \text{ Verifier has a winning strategy in the game } \mathcal{G}_\infty(\mathfrak{A}, s, \phi).$$

  We call $\mathcal{G}_\infty$ the **unbounded evaluation game**.
- Positions of the game are triples $(r, \psi, \#)$, where r is the current assignment, ψ is a subformula of ϕ and $\# \in \{-, +\}$. Initial position is just $(s, \phi, +)$.
- Rules for first-order connectives and atomic formulas are standard. Some examples.
  - Next position from $(r, \neg\psi, +)$ is $(r, \psi, -)$ and from $(r, \neg\psi, -)$ the next position is $(r, \psi, +)$.

Game-theoretical semantics (GTS) for SCL

- We associate to each τ-structure $\mathfrak{A}$, assignment s and a formula ϕ of $\mathrm{SCL}[\tau]$ a two-player game $\mathcal{G}_\infty(\mathfrak{A}, s, \phi)$, which is played by Verifier and Falsifier. We then define that

  $$\mathfrak{A}, s \models \phi \Leftrightarrow \text{ Verifier has a winning strategy in the game } \mathcal{G}_\infty(\mathfrak{A}, s, \phi).$$

  We call $\mathcal{G}_\infty$ the **unbounded evaluation game**.
- Positions of the game are triples $(r, \psi, \#)$, where r is the current assignment, ψ is a subformula of ϕ and $\# \in \{-, +\}$. Initial position is just $(s, \phi, +)$.
- Rules for first-order connectives and atomic formulas are standard. Some examples.
  - Next position from $(r, \neg\psi, +)$ is $(r, \psi, -)$ and from $(r, \neg\psi, -)$ the next position is $(r, \psi, +)$.
  - If the position is $(r, \alpha, +)$, where α is an atomic formula, then Verifier wins if $\mathfrak{A}, r \models \alpha$ and otherwise Falsifier wins.

Game-theoretical semantics (GTS) for SCL

- Next position from $(r, L\psi, \#)$ is $(r, \psi, \#)$.

Game-theoretical semantics (GTS) for SCL

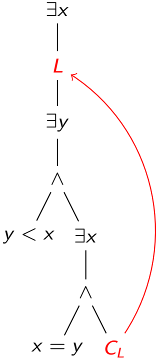
- Next position from $(r, L\psi, \#)$ is $(r, \psi, \#)$.
- Next position from $(r, C_L, \#)$ is $(r, \mathrm{Rf}(C_L), \#)$, where $\mathrm{Rf}(C_L)$ is the **reference formula** of C_L.

Game-theoretical semantics (GTS) for SCL

- Next position from $(r, L\psi, \#)$ is $(r, \psi, \#)$.
- Next position from $(r, C_L, \#)$ is $(r, \mathrm{Rf}(C_L), \#)$, where $\mathrm{Rf}(C_L)$ is the **reference formula** of C_L. It is defined as the subformula occurrence $L\psi$ such that there is a directed path from $L\psi$ to C_L in the syntax tree of ϕ, and L does not occur strictly between $L\psi$ and C_L on that path.

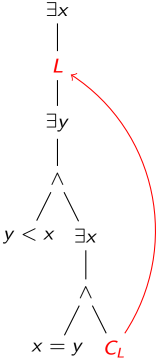
Game-theoretical semantics (GTS) for SCL

- Next position from $(r, L\psi, \#)$ is $(r, \psi, \#)$.
- Next position from $(r, C_L, \#)$ is $(r, \mathrm{Rf}(C_L), \#)$, where $\mathrm{Rf}(C_L)$ is the **reference formula** of C_L. It is defined as the subformula occurrence $L\psi$ such that there is a directed path from $L\psi$ to C_L in the syntax tree of ϕ, and L does not occur strictly between $L\psi$ and C_L on that path.



Game-theoretical semantics (GTS) for SCL

- Next position from $(r, L\psi, \#)$ is $(r, \psi, \#)$.
- Next position from $(r, C_L, \#)$ is $(r, \mathrm{Rf}(C_L), \#)$, where $\mathrm{Rf}(C_L)$ is the **reference formula** of C_L. It is defined as the subformula occurrence $L\psi$ such that there is a directed path from $L\psi$ to C_L in the syntax tree of ϕ, and L does not occur strictly between $L\psi$ and C_L on that path.



- Neither player wins infinitely long plays.

Bounded SCL

- Bounded SCL (BndSCL) is obtained from SCL by replacing the unbounded evaluation game $\mathcal{G}_\infty$ with the **bounded** evaluation game $\mathcal{G}_\omega$.

Bounded SCL

- Bounded SCL (BndSCL) is obtained from SCL by replacing the unbounded evaluation game $\mathcal{G}_\infty$ with the **bounded** evaluation game $\mathcal{G}_\omega$.
- Given a triple $(\mathfrak{A}, s, \phi)$, the game $\mathcal{G}_\omega(\mathfrak{A}, s, \phi)$ proceeds as follows.

Bounded SCL

- Bounded SCL (BndSCL) is obtained from SCL by replacing the unbounded evaluation game $\mathcal{G}_\infty$ with the **bounded** evaluation game $\mathcal{G}_\omega$.
- Given a triple $(\mathfrak{A}, s, \phi)$, the game $\mathcal{G}_\omega(\mathfrak{A}, s, \phi)$ proceeds as follows.
  1. Falsifier chooses a natural number $n' \in \mathbb{N}$.

Bounded SCL

- Bounded SCL (BndSCL) is obtained from SCL by replacing the unbounded evaluation game $\mathcal{G}_\infty$ with the **bounded** evaluation game $\mathcal{G}_\omega$.
- Given a triple $(\mathfrak{A}, s, \phi)$, the game $\mathcal{G}_\omega(\mathfrak{A}, s, \phi)$ proceeds as follows.
  1. Falsifier chooses a natural number $n' \in \mathbb{N}$.
  2. Verifier chooses a natural number $n \geq n'$.

Bounded SCL

- Bounded SCL (BndSCL) is obtained from SCL by replacing the unbounded evaluation game $\mathcal{G}_\infty$ with the **bounded** evaluation game $\mathcal{G}_\omega$.
- Given a triple $(\mathfrak{A}, s, \phi)$, the game $\mathcal{G}_\omega(\mathfrak{A}, s, \phi)$ proceeds as follows.
    1. Falsifier chooses a natural number $n' \in \mathbb{N}$.
    2. Verifier chooses a natural number $n \geq n'$.
    3. Players play the game n-**bounded** evaluation game $\mathcal{G}_n(\mathfrak{A}, s, \phi)$.

Bounded SCL

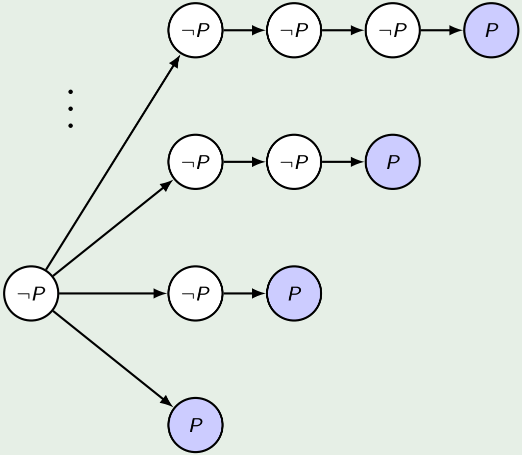
- Bounded SCL (BndSCL) is obtained from SCL by replacing the unbounded evaluation game $\mathcal{G}_\infty$ with the **bounded** evaluation game $\mathcal{G}_\omega$.
- Given a triple $(\mathfrak{A}, s, \phi)$, the game $\mathcal{G}_\omega(\mathfrak{A}, s, \phi)$ proceeds as follows.
    1. Falsifier chooses a natural number $n' \in \mathbb{N}$.
    2. Verifier chooses a natural number $n \geq n'$.
    3. Players play the game n-**bounded** evaluation game $\mathcal{G}_n(\mathfrak{A}, s, \phi)$.
- The n-bounded evaluation game $\mathcal{G}_n$ works like $\mathcal{G}_\infty$ with the exception that looping atoms can be visited at most n times.

Bounded SCL

- Bounded SCL (BndSCL) is obtained from SCL by replacing the unbounded evaluation game $\mathcal{G}_\infty$ with the **bounded** evaluation game $\mathcal{G}_\omega$.
- Given a triple $(\mathfrak{A}, s, \phi)$, the game $\mathcal{G}_\omega(\mathfrak{A}, s, \phi)$ proceeds as follows.
  1. Falsifier chooses a natural number $n' \in \mathbb{N}$.
  2. Verifier chooses a natural number $n \geq n'$.
  3. Players play the game n-**bounded** evaluation game $\mathcal{G}_n(\mathfrak{A}, s, \phi)$.
- The n-bounded evaluation game $\mathcal{G}_n$ works like $\mathcal{G}_\infty$ with the exception that looping atoms can be visited at most n times. If the players reach a looping atom after they have visited looping atoms n times, the game stops and neither player wins the game.

BndSCL vs SCL

Example



Consider the formula $\phi(x) := L(P(x) \vee \forall y(R(x, y) \rightarrow \exists x(x = y \wedge C_L)))$. Under unbounded semantics, ϕ is true at the "root" of the above structure, while under bounded semantics it is not true.

Approximants

Definition

Let ϕ be a formula of SCL.

Approximants

Definition

Let ϕ be a formula of SCL. The ***n*th unfolding** of ϕ, denoted by Ψ_{ϕ}^{n}, is defined inductively as follows.

Approximants

Definition

Let ϕ be a formula of SCL. The ***n*th unfolding** of ϕ, denoted by Ψ_{ϕ}^{n}, is defined inductively as follows.

1. The zeroeth unfolding Ψ_{ϕ}^{0} of ϕ is defined to be the formula ϕ.

Approximants

Definition

Let ϕ be a formula of SCL. The ***n*th unfolding** of ϕ, denoted by Ψ^n_ϕ, is defined inductively as follows.

1. The zeroeth unfolding Ψ^0_ϕ of ϕ is defined to be the formula ϕ.
2. The $(k+1)$st unfolding Ψ^{k+1}_ϕ is the formula obtained from the kth unfolding Ψ^k_ϕ by replacing every looping atom C_L in Ψ^k_ϕ by the corresponding reference formula $\mathrm{Rf}(C_L)$ in Ψ^k_ϕ.

Approximants

Definition

Let ϕ be a formula of SCL. The ***n*th unfolding** of ϕ, denoted by Ψ_{ϕ}^{n}, is defined inductively as follows.

1. The zeroeth unfolding Ψ_{ϕ}^{0} of ϕ is defined to be the formula ϕ.
2. The $(k+1)$st unfolding Ψ_{ϕ}^{k+1} is the formula obtained from the kth unfolding Ψ_{ϕ}^{k} by replacing every looping atom C_L in Ψ_{ϕ}^{k} by the corresponding reference formula $\mathrm{Rf}(C_L)$ in Ψ_{ϕ}^{k}.

Example

Consider the formula $\phi := \exists x L \exists y (R(x, y) \wedge C_L)$.

Approximants

Definition

Let ϕ be a formula of SCL. The ***n*th unfolding** of ϕ, denoted by Ψ_ϕ^n, is defined inductively as follows.

1. The zeroeth unfolding Ψ_ϕ^0 of ϕ is defined to be the formula ϕ.
2. The $(k+1)$st unfolding Ψ_ϕ^{k+1} is the formula obtained from the kth unfolding Ψ_ϕ^k by replacing every looping atom C_L in Ψ_ϕ^k by the corresponding reference formula $\mathrm{Rf}(C_L)$ in Ψ_ϕ^k.

Example

Consider the formula $\phi := \exists x L \exists y (R(x,y) \wedge C_L)$.

- $\Psi_\phi^0 := \exists x L \exists y (R(x,y) \wedge C_L)$.

Approximants

Definition

Let ϕ be a formula of SCL. The ***n*th unfolding** of ϕ, denoted by Ψ_ϕ^n, is defined inductively as follows.

1. The zeroeth unfolding Ψ_ϕ^0 of ϕ is defined to be the formula ϕ.
2. The $(k+1)$st unfolding Ψ_ϕ^{k+1} is the formula obtained from the kth unfolding Ψ_ϕ^k by replacing every looping atom C_L in Ψ_ϕ^k by the corresponding reference formula $\mathrm{Rf}(C_L)$ in Ψ_ϕ^k.

Example

Consider the formula $\phi := \exists x L \exists y (R(x,y) \wedge C_L)$.

- $\Psi_\phi^0 := \exists x L \exists y (R(x,y) \wedge C_L)$.
- $\Psi_\phi^1 := \exists x L \exists y (R(x,y) \wedge L \exists y (R(x,y) \wedge C_L))$

Approximants

Definition

Let ϕ be a formula of SCL. The ***n*th unfolding** of ϕ, denoted by Ψ^n_ϕ, is defined inductively as follows.

1. The zeroeth unfolding Ψ^0_ϕ of ϕ is defined to be the formula ϕ.
2. The $(k+1)$st unfolding Ψ^{k+1}_ϕ is the formula obtained from the kth unfolding Ψ^k_ϕ by replacing every looping atom C_L in Ψ^k_ϕ by the corresponding reference formula $\mathrm{Rf}(C_L)$ in Ψ^k_ϕ.

Example

Consider the formula $\phi := \exists x L \exists y(R(x,y) \wedge C_L)$.

- $\Psi^0_\phi := \exists x L \exists y(R(x,y) \wedge C_L)$.
- $\Psi^1_\phi := \exists x L \exists y(R(x,y) \wedge L \exists y(R(x,y) \wedge C_L))$
- $\Psi^2_\phi := \exists x L \exists y(R(x,y) \wedge L \exists y(R(x,y) \wedge L \exists y(R(x,y) \wedge C_L)))$

Approximants

Definition

Let ϕ be a formula of SCL.

Approximants

Definition

Let ϕ be a formula of SCL. We define the ***n*th approximant** (or ***n*-approximant**) Φ_ϕ^n of ϕ to be the FO-formula obtained from the nth unfolding Ψ_ϕ^n by removing all the label symbols and replacing each occurrence of each looping atom by

Approximants

Definition

Let ϕ be a formula of SCL. We define the ***n*th approximant** (or ***n*-approximant**) Φ^n_ϕ of ϕ to be the FO-formula obtained from the *n*th unfolding Ψ^n_ϕ by removing all the label symbols and replacing each occurrence of each looping atom by

1. $\perp$ if the occurrence of the atom is positive in Ψ^n_ϕ,

Approximants

Definition

Let ϕ be a formula of SCL. We define the ***n*th approximant** (or ***n*-approximant**) Φ^n_ϕ of ϕ to be the FO-formula obtained from the *n*th unfolding Ψ^n_ϕ by removing all the label symbols and replacing each occurrence of each looping atom by

1. $\bot$ if the occurrence of the atom is positive in Ψ^n_ϕ,
2. $\top$ if the occurrence is negative in Ψ^n_ϕ.

Approximants

Definition

Let ϕ be a formula of SCL. We define the ***n*th approximant** (or ***n*-approximant**) Φ^n_ϕ of ϕ to be the FO-formula obtained from the nth unfolding Ψ^n_ϕ by removing all the label symbols and replacing each occurrence of each looping atom by

1. $\perp$ if the occurrence of the atom is positive in Ψ^n_ϕ,
2. $\top$ if the occurrence is negative in Ψ^n_ϕ.

Example

Recall the formula $\phi := \exists x L \exists y (R(x, y) \wedge C_L)$.

Approximants

Definition

Let ϕ be a formula of SCL. We define the ***n*th approximant** (or ***n*-approximant**) Φ^n_ϕ of ϕ to be the FO-formula obtained from the nth unfolding Ψ^n_ϕ by removing all the label symbols and replacing each occurrence of each looping atom by

1. $\bot$ if the occurrence of the atom is positive in Ψ^n_ϕ,
2. $\top$ if the occurrence is negative in Ψ^n_ϕ.

Example

Recall the formula $\phi := \exists x L \exists y(R(x,y) \wedge C_L)$. Since

$$\Psi^2_\phi := \exists x L \exists y(R(x,y) \wedge L \exists y(R(x,y) \wedge L \exists y(R(x,y) \wedge C_L)))$$

Approximants

Definition

Let ϕ be a formula of SCL. We define the ***n*th approximant** (or ***n*-approximant**) Φ^n_ϕ of ϕ to be the FO-formula obtained from the nth unfolding Ψ^n_ϕ by removing all the label symbols and replacing each occurrence of each looping atom by

1. $\perp$ if the occurrence of the atom is positive in Ψ^n_ϕ,
2. $\top$ if the occurrence is negative in Ψ^n_ϕ.

Example

Recall the formula $\phi := \exists x L \exists y (R(x,y) \wedge C_L)$. Since

$$\Psi^2_\phi := \exists x L \exists y (R(x,y) \wedge L \exists y (R(x,y) \wedge L \exists y (R(x,y) \wedge C_L)))$$

we have that

$$\Phi^2_\phi := \exists x \exists y (R(x,y) \wedge \exists y (R(x,y) \wedge \exists y (R(x,y) \wedge \perp)))$$

BndSCL as a fragment of $\mathcal{L}^{\omega}_{\omega_1\omega}$

Lemma

Let ϕ be a formula of SCL.

BndSCL as a fragment of $\mathcal{L}^{\omega}_{\omega_1\omega}$

Lemma

Let ϕ be a formula of SCL. Then for every structure $\mathfrak{A}$ and assignment s we have that

$$\text{Verifier has a winning strategy in } \mathcal{G}_n(\mathfrak{A}, s, \phi) \Leftrightarrow \mathfrak{A}, s \models \Phi^n_\phi$$

BndSCL as a fragment of $\mathcal{L}^{\omega}_{\omega_1\omega}$

Lemma

Let ϕ be a formula of SCL. Then for every structure $\mathfrak{A}$ and assignment s we have that

$$\textit{Verifier has a winning strategy in } \mathcal{G}_n(\mathfrak{A}, s, \phi) \Leftrightarrow \mathfrak{A}, s \models \Phi^n_\phi$$

Theorem

Let ϕ be a formula of BndSCL.

BndSCL as a fragment of $\mathcal{L}^{\omega}_{\omega_1\omega}$

Lemma

Let ϕ be a formula of SCL. Then for every structure $\mathfrak{A}$ and assignment s we have that

$$\text{Verifier has a winning strategy in } \mathcal{G}_n(\mathfrak{A}, s, \phi) \Leftrightarrow \mathfrak{A}, s \models \Phi^n_\phi$$

Theorem

Let ϕ be a formula of BndSCL. Then for every structure $\mathfrak{A}$ and assignment s we have that

$$\text{Verifier has a winning strategy in } \mathcal{G}_\omega(\mathfrak{A}, s, \phi) \Leftrightarrow \mathfrak{A}, s \models \bigvee_{n \in \mathbb{N}} \Phi^n_\phi$$

BndSCL as a fragment of $\mathcal{L}^{\omega}_{\omega_1\omega}$

Lemma

Let ϕ be a formula of SCL. Then for every structure $\mathfrak{A}$ and assignment s we have that

$$\text{Verifier has a winning strategy in } \mathcal{G}_n(\mathfrak{A}, s, \phi) \Leftrightarrow \mathfrak{A}, s \models \Phi^n_\phi$$

Theorem

Let ϕ be a formula of BndSCL. Then for every structure $\mathfrak{A}$ and assignment s we have that

$$\text{Verifier has a winning strategy in } \mathcal{G}_\omega(\mathfrak{A}, s, \phi) \Leftrightarrow \mathfrak{A}, s \models \bigvee_{n\in\mathbb{N}} \Phi^n_\phi$$

In particular, BndSCL $\leq \mathcal{L}^{\omega}_{\omega_1\omega}$.

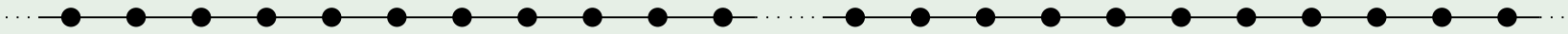
SCL $\not\leq$ BndSCL

Example

Connectivity of a graph is not definable in BndSCL. For example, consider the following graphs

$\mathfrak{G}_1$:

$\mathfrak{G}_2$:



These graphs are elementary equivalent, which implies that for every BndSCL sentence ϕ we have that if $\mathfrak{G}_1 \models \phi$, then $\mathfrak{G}_2 \models \phi$.

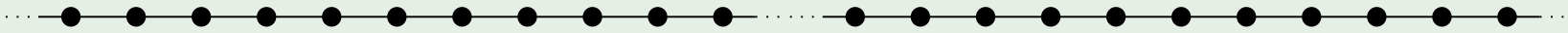
SCL $\not\leq$ BndSCL

Example

Connectivity of a graph is not definable in BndSCL. For example, consider the following graphs

$\mathfrak{G}_1$:

$\mathfrak{G}_2$:



These graphs are elementary equivalent, which implies that for every BndSCL sentence ϕ we have that if $\mathfrak{G}_1 \models \phi$, then $\mathfrak{G}_2 \models \phi$.

Problem

Is BndSCL contained in SCL?

Valid sentences of BndSCL are RE

Theorem

Let ϕ be a sentence of BndSCL. Now ϕ is valid if and only if Φ_ϕ^n is valid, for some $n \in \mathbb{N}$.

Valid sentences of BndSCL are RE

Theorem

Let ϕ be a sentence of BndSCL. Now ϕ is valid if and only if Φ^n_ϕ is valid, for some $n \in \mathbb{N}$.

Proof.

($\Leftarrow$) If Verifier has a winning strategy in $\mathcal{G}_n(\mathfrak{A}, \phi)$, then they have a winning strategy in $\mathcal{G}_\omega(\mathfrak{A}, \phi)$.

Valid sentences of BndSCL are RE

Theorem

Let ϕ be a sentence of BndSCL. Now ϕ is valid if and only if Φ^n_ϕ is valid, for some $n \in \mathbb{N}$.

Proof.

($\Leftarrow$) If Verifier has a winning strategy in $\mathcal{G}_n(\mathfrak{A}, \phi)$, then they have a winning strategy in $\mathcal{G}_\omega(\mathfrak{A}, \phi)$.

($\Rightarrow$) Suppose that $\neg\Phi^n_\phi$ is satisfiable, for every $n \in \mathbb{N}$. Since $\neg\Phi^n_\phi \models \neg\Phi^{n'}_\phi$, for every $n' < n$, using compactness we get that $\{\neg\Phi^n_\phi \mid n \in \mathbb{N}\}$ is satisfiable. Since ϕ is truth equivalent with $\bigvee_{n\in\mathbb{N}} \Phi^n_\phi$, there must exists a model $\mathfrak{A}$ such that Verifier does not have a winning strategy in $\mathcal{G}_\omega(\mathfrak{A}, \phi)$. $\square$

Valid sentences of SCL are RE

Theorem

Let ϕ be a sentence of SCL. Now ϕ is valid if and only if Φ_ϕ^n is valid, for some $n \in \mathbb{N}$.

Valid sentences of SCL are RE

Theorem

Let ϕ be a sentence of SCL. Now ϕ is valid if and only if Φ_{ϕ}^{n} is valid, for some $n \in \mathbb{N}$.

Proof.

($\Leftarrow$) If Verifier has a winning strategy in $\mathcal{G}_n(\mathfrak{A}, \phi)$, then they have a winning strategy in $\mathcal{G}_\infty(\mathfrak{A}, \phi)$.

Valid sentences of SCL are RE

Theorem

Let ϕ be a sentence of SCL. Now ϕ is valid if and only if Φ^n_ϕ is valid, for some $n \in \mathbb{N}$.

Proof.

($\Leftarrow$) If Verifier has a winning strategy in $\mathcal{G}_n(\mathfrak{A}, \phi)$, then they have a winning strategy in $\mathcal{G}_\infty(\mathfrak{A}, \phi)$.

($\Rightarrow$) If there exists for every $n \in \mathbb{N}$ a structure $\mathfrak{A}_n$ such that Verifier does not have a winning strategy in $\mathcal{G}_n(\mathfrak{A}_n, \phi)$, then one can use compactness theorem for FO to construct a structure $\mathfrak{A}$ such that Verifier does not have a winning strategy in $\mathcal{G}_\infty(\mathfrak{A}, \phi)$. $\square$

Valid sentences of SCL are RE

Theorem

Let ϕ be a sentence of SCL. Now ϕ is valid if and only if Φ^n_ϕ is valid, for some $n \in \mathbb{N}$.

Proof.

($\Leftarrow$) If Verifier has a winning strategy in $\mathcal{G}_n(\mathfrak{A}, \phi)$, then they have a winning strategy in $\mathcal{G}_\infty(\mathfrak{A}, \phi)$.

($\Rightarrow$) If there exists for every $n \in \mathbb{N}$ a structure $\mathfrak{A}_n$ such that Verifier does not have a winning strategy in $\mathcal{G}_n(\mathfrak{A}_n, \phi)$, then one can use compactness theorem for FO to construct a structure $\mathfrak{A}$ such that Verifier does not have a winning strategy in $\mathcal{G}_\infty(\mathfrak{A}, \phi)$. $\square$

Corollary

Valid sentences of BndSCL and SCL coincide.

Weakly complete axiomatization for SCL

- In our work we also developed a natural deduction style proof system which is weakly complete: if Σ is a set of FO-sentences and ϕ is a sentence of SCL, then $\Sigma \models \phi$ iff $\Sigma \vdash \phi$ in our system.

Weakly complete axiomatization for SCL

- In our work we also developed a natural deduction style proof system which is weakly complete: if Σ is a set of FO-sentences and ϕ is a sentence of SCL, then $\Sigma \models \phi$ iff $\Sigma \vdash \phi$ in our system.

$$\frac{\varphi[L\psi]}{\varphi[L\psi\{L\psi/C_L\}]} \updownarrow \text{(}L\text{Subst1)} \qquad \frac{\varphi[L\psi]}{\varphi[L\psi\{\psi/C_L\}]} \updownarrow \text{(}L\text{Subst2)}$$

$$\frac{\varphi[L\psi]}{\varphi[L'L\psi\{\neg C_{L'}/\neg C_L\}]} \updownarrow \text{(}L\text{Dual-Intro)} \qquad \frac{\varphi[\psi]}{\varphi[L\psi]} \updownarrow \text{(}L\text{Dummy-Intro-Elim)}$$

$$\frac{\varphi[L\psi]}{\varphi[L'\psi\{\{C_{L'}/C_L\}\}]} \updownarrow \text{(}LC_L\text{Rename)} \qquad \frac{\varphi[C_L]}{\varphi[\psi/C_L]} \text{(}C_L\text{Free-Elim)}$$

Weakly complete axiomatization for SCL

- In our work we also developed a natural deduction style proof system which is weakly complete: if Σ is a set of FO-sentences and ϕ is a sentence of SCL, then $\Sigma \models \phi$ iff $\Sigma \vdash \phi$ in our system.

$$\frac{\varphi[L\psi]}{\varphi[L\psi\{L\psi/C_L\}]}\updownarrow \text{(LSubst1)} \qquad \frac{\varphi[L\psi]}{\varphi[L\psi\{\psi/C_L\}]}\updownarrow \text{(LSubst2)}$$

$$\frac{\varphi[L\psi]}{\varphi[L'L\psi\{\neg C_{L'}/\neg C_L\}]}\updownarrow \text{(LDual-Intro)} \qquad \frac{\varphi[\psi]}{\varphi[L\psi]}\updownarrow \text{(LDummy-Intro-Elim)}$$

$$\frac{\varphi[L\psi]}{\varphi[L'\psi\{\{C_{L'}/C_L\}\}]}\updownarrow \text{(}LC_L\text{ Rename)} \qquad \frac{\varphi[C_L]}{\varphi[\psi/C_L]} \text{(}C_L\text{ Free-Elim)}$$

- The main idea is to show that for each formula ϕ we have that $\Phi^n_\phi \vdash \phi$.

Weakly complete axiomatization for SCL

- In our work we also developed a natural deduction style proof system which is weakly complete: if Σ is a set of FO-sentences and ϕ is a sentence of SCL, then $\Sigma \models \phi$ iff $\Sigma \vdash \phi$ in our system.

$$\frac{\varphi[L\psi]}{\varphi[L\psi\{L\psi/C_L\}]}\updownarrow \text{(LSubst1)} \qquad \frac{\varphi[L\psi]}{\varphi[L\psi\{\psi/C_L\}]}\updownarrow \text{(LSubst2)}$$

$$\frac{\varphi[L\psi]}{\varphi[L'L\psi\{\neg C_{L'}/\neg C_L\}]}\updownarrow \text{(LDual-Intro)} \qquad \frac{\varphi[\psi]}{\varphi[L\psi]}\updownarrow \text{(LDummy-Intro-Elim)}$$

$$\frac{\varphi[L\psi]}{\varphi[L'\psi\{\{C_{L'}/C_L\}\}]}\updownarrow \text{(}LC_L\text{ Rename)} \qquad \frac{\varphi[C_L]}{\varphi[\psi/C_L]} \text{(}C_L\text{ Free-Elim)}$$

- The main idea is to show that for each formula ϕ we have that $\Phi^n_\phi \vdash \phi$.
- As an important step of our proof we show that our system can prove that every SCL sentence is equivalent to a sentence in **strong negation normal form**: negation only occurs in front of atomic FO-formulas.

Validity problem for SCL^2

Theorem

For every sentence ϕ *of* SCL^k *there exists a sentence* Ψ *of* ESO^k *such that for every structure* $\mathfrak{A}$ *we have the following equivalence*

$$\text{Verifier does not have a winning strategy in } \mathcal{G}_\infty(\mathfrak{A}, \phi) \Leftrightarrow \mathfrak{A} \models \Psi$$

Validity problem for SCL^2

Theorem

For every sentence ϕ of SCL^k there exists a sentence Ψ of ESO^k such that for every structure $\mathfrak{A}$ we have the following equivalence

$$\text{Verifier does not have a winning strategy in } \mathcal{G}_\infty(\mathfrak{A}, \phi) \Leftrightarrow \mathfrak{A} \models \Psi$$

Furthermore, Ψ can be computed from ϕ in polynomial time

Validity problem for SCL^2

Theorem

For every sentence ϕ of SCL^k there exists a sentence Ψ of ESO^k such that for every structure $\mathfrak{A}$ we have the following equivalence

$$\text{Verifier does not have a winning strategy in } \mathcal{G}_\infty(\mathfrak{A}, \phi) \Leftrightarrow \mathfrak{A} \models \Psi$$

Furthermore, Ψ can be computed from ϕ in polynomial time

Corollary

1. *Every sentence of SCL^k can be translated in polynomial time to an equivalent sentence of $\forall\mathrm{SO}^k$.*

Validity problem for SCL^2

Theorem

For every sentence ϕ of SCL^k there exists a sentence Ψ of ESO^k such that for every structure $\mathfrak{A}$ we have the following equivalence

$$\textit{Verifier does not have a winning strategy in } \mathcal{G}_\infty(\mathfrak{A}, \phi) \Leftrightarrow \mathfrak{A} \models \Psi$$

Furthermore, Ψ can be computed from ϕ in polynomial time

Corollary

1. *Every sentence of SCL^k can be translated in polynomial time to an equivalent sentence of $\forall\mathrm{SO}^k$.*
2. *The validity problem for SCL^2 is* coNExpTime*-complete.*

Validity problem for SCL^2

Theorem

For every sentence ϕ of SCL^k there exists a sentence Ψ of ESO^k such that for every structure $\mathfrak{A}$ we have the following equivalence

$$\textit{Verifier does not have a winning strategy in } \mathcal{G}_\infty(\mathfrak{A}, \phi) \Leftrightarrow \mathfrak{A} \models \Psi$$

Furthermore, Ψ can be computed from ϕ in polynomial time

Corollary

1. *Every sentence of SCL^k can be translated in polynomial time to an equivalent sentence of $\forall\mathrm{SO}^k$.*
2. *The validity problem for SCL^2 is* coNEXPTIME*-complete.*

Problem

Are the satisfiability problems of BndSCL^2 and SCL^2 decidable? BndSCL^2 has the finite model property and SCL^2 does not have it.

Downward Löwenheim–Skolem Theorem

Theorem

Let ϕ be a sentence of BndSCL or SCL and let $\mathfrak{A}$ be a model of ϕ.

Downward Löwenheim–Skolem Theorem

Theorem

Let ϕ be a sentence of BndSCL or SCL and let $\mathfrak{A}$ be a model of ϕ. Then there exists a countable substructure $\mathfrak{B}$ of $\mathfrak{A}$ such that $\mathfrak{B} \models \phi$.

Downward Löwenheim–Skolem Theorem

Theorem

Let ϕ be a sentence of BndSCL or SCL and let $\mathfrak{A}$ be a model of ϕ. Then there exists a countable substructure $\mathfrak{B}$ of $\mathfrak{A}$ such that $\mathfrak{B} \models \phi$.

Proof.

Proof for SCL.

Downward Löwenheim–Skolem Theorem

Theorem

Let ϕ be a sentence of BndSCL or SCL and let $\mathfrak{A}$ be a model of ϕ. Then there exists a countable substructure $\mathfrak{B}$ of $\mathfrak{A}$ such that $\mathfrak{B} \models \phi$.

Proof.

Proof for SCL. Fix a (positional) winning strategy σ for the Verifier in the game $\mathcal{G}_\infty(\mathfrak{A}, \phi)$. We want to construct a countable model $\mathfrak{B}$ so that Eloise has a winning strategy also in the game $\mathcal{G}_\infty(\mathfrak{B}, \phi)$.

Downward Löwenheim–Skolem Theorem

Theorem

Let ϕ be a sentence of BndSCL or SCL and let $\mathfrak{A}$ be a model of ϕ. Then there exists a countable substructure $\mathfrak{B}$ of $\mathfrak{A}$ such that $\mathfrak{B} \models \phi$.

Proof.

Proof for SCL. Fix a (positional) winning strategy σ for the Verifier in the game $\mathcal{G}_\infty(\mathfrak{A}, \phi)$. We want to construct a countable model $\mathfrak{B}$ so that Eloise has a winning strategy also in the game $\mathcal{G}_\infty(\mathfrak{B}, \phi)$. Pick an arbitrary $b \in A$. We define a sequence of sets $(B_n)_{n\in\mathbb{N}}$ inductively such that, firstly $B_0 = \{b\}$, and then

$$B_{n+1} = B_n \cup \{d \mid \sigma((\exists x\psi, s, +)) = d\},$$

where $\mathrm{range}(s) \subseteq B_n$ and $\exists x\psi \in \mathrm{Subf}(\phi)$. Let $\mathfrak{B}$ be the substructure of $\mathfrak{A}$ induced by the set $\bigcup_{n\in\mathbb{N}} B_n$. $\mathfrak{B}$ is clearly countable. □

Failure of Craig interpolation

- Craig interpolation property (CIP): if $\phi \models \psi$, then there exists a third sentence θ such that $\phi \models \theta \models \psi$ and θ contains only those relation symbols that occur in both of the sentences ϕ and ψ.

Failure of Craig interpolation

- Craig interpolation property (CIP): if $\phi \models \psi$, then there exists a third sentence θ such that $\phi \models \theta \models \psi$ and θ contains only those relation symbols that occur in both of the sentences ϕ and ψ.
- Neither BndSCL nor SCL has CIP.

Failure of Craig interpolation

- Craig interpolation property (CIP): if $\phi \models \psi$, then there exists a third sentence θ such that $\phi \models \theta \models \psi$ and θ contains only those relation symbols that occur in both of the sentences ϕ and ψ.
- Neither BndSCL nor SCL has CIP.

Lemma

For every sentence ϕ of SCL$[\varnothing]$ there exists a finite structure $\mathfrak{A}$ of even size and a finite structure $\mathfrak{B}$ of odd size such that

$$\mathfrak{A} \models \phi \Rightarrow \mathfrak{B} \models \phi$$

Failure of Craig interpolation

- Craig interpolation property (CIP): if $\phi \models \psi$, then there exists a third sentence θ such that $\phi \models \theta \models \psi$ and θ contains only those relation symbols that occur in both of the sentences ϕ and ψ.
- Neither BndSCL nor SCL has CIP.

Lemma

For every sentence ϕ of SCL$[\varnothing]$ there exists a finite structure $\mathfrak{A}$ of even size and a finite structure $\mathfrak{B}$ of odd size such that

$$\mathfrak{A} \models \phi \Rightarrow \mathfrak{B} \models \phi$$

Proof.

Follows from the fact that over finite models SCL is contained in $\mathcal{L}^{\omega}_{\omega_1\omega}$. $\square$

Failure of Craig interpolation

Theorem

SCL does not have CIP.

Failure of Craig interpolation

Theorem

SCL does not have CIP.

Proof.

Given a binary relation "$<$", there is a SCL sentence ϕ which states that

1. $<$ is a (total) linear order
2. the distance between the smallest and the largest element is finite.

In particular, ϕ projectively defines the class of finite models.

Failure of Craig interpolation

Theorem

SCL does not have CIP.

Proof.

Given a binary relation "$<$", there is a SCL sentence ϕ which states that

1. $<$ is a (total) linear order
2. the distance between the smallest and the largest element is finite.

In particular, ϕ projectively defines the class of finite models.

Fix two binary relation symbols E_1 and E_2. It is easy to write sentences $\psi_1 \in \mathrm{FO}[\{E_1\}]$ and $\psi_2 \in \mathrm{FO}[\{E_2\}]$ such that

Failure of Craig interpolation

Theorem

SCL does not have CIP.

Proof.

Given a binary relation "$<$", there is a SCL sentence ϕ which states that

1. $<$ is a (total) linear order
2. the distance between the smallest and the largest element is finite.

In particular, ϕ projectively defines the class of finite models.

Fix two binary relation symbols E_1 and E_2. It is easy to write sentences $\psi_1 \in \mathrm{FO}[\{E_1\}]$ and $\psi_2 \in \mathrm{FO}[\{E_2\}]$ such that

1. ψ_1 states that E_1 is an equivalence relation where each equivalence class has cardinality two

Failure of Craig interpolation

Theorem

SCL does not have CIP.

Proof.

Given a binary relation "$<$", there is a SCL sentence ϕ which states that

1. $<$ is a (total) linear order
2. the distance between the smallest and the largest element is finite.

In particular, ϕ projectively defines the class of finite models.

Fix two binary relation symbols E_1 and E_2. It is easy to write sentences $\psi_1 \in \mathrm{FO}[\{E_1\}]$ and $\psi_2 \in \mathrm{FO}[\{E_2\}]$ such that

1. ψ_1 states that E_1 is an equivalence relation where each equivalence class has cardinality two
2. ψ_2 states that E_2 is an equivalence relation with one class of cardinality one while each other class has cardinality two.

Failure of Craig interpolation

Theorem

SCL does not have CIP.

Proof.

Given a binary relation "$<$", there is a SCL sentence ϕ which states that

1. $<$ is a (total) linear order
2. the distance between the smallest and the largest element is finite.

In particular, ϕ projectively defines the class of finite models.

Fix two binary relation symbols E_1 and E_2. It is easy to write sentences $\psi_1 \in \mathrm{FO}[\{E_1\}]$ and $\psi_2 \in \mathrm{FO}[\{E_2\}]$ such that

1. ψ_1 states that E_1 is an equivalence relation where each equivalence class has cardinality two
2. ψ_2 states that E_2 is an equivalence relation with one class of cardinality one while each other class has cardinality two.

Clearly $\phi \wedge \psi_1 \models \neg\psi_2$. However, any interpolant between these sentences needs to distinguish each finite structure of even size from every finite structure of odd size. □

Everywhere determined sentences

Theorem

Let ϕ be a sentence of SCL. If ϕ is determined everywhere, then there exists $n \in \mathbb{N}$ such that ϕ is equivalent with Φ^n_ϕ. In particular, any sentence of SCL which expresses a property that is not FO*-definable is undetermined in some model.*

Everywhere determined sentences

Theorem

Let ϕ be a sentence of SCL. If ϕ is determined everywhere, then there exists $n \in \mathbb{N}$ such that ϕ is equivalent with Φ^n_ϕ. In particular, any sentence of SCL which expresses a property that is not FO-definable is undetermined in some model.

Proof.

If ϕ is determined everywhere, then $\phi \vee \neg\phi$ is valid. This in turn implies that $\Phi^n_{\phi\vee\neg\phi} = \Phi^n_\phi \vee \Phi^n_{\neg\phi}$ is also valid, for some $n \in \mathbb{N}$. We claim that ϕ is equivalent with Φ^n_ϕ. First, we have that $\Phi^n_\phi \models \phi$. Secondly, since $\Phi^n_\phi \vee \Phi^n_{\neg\phi}$ is valid, we have that $\neg\Phi^n_\phi \models \Phi^n_{\neg\phi} \models \neg\phi$. □

Everywhere determined sentences

Theorem

Let ϕ be a sentence of SCL. If ϕ is determined everywhere, then there exists $n \in \mathbb{N}$ such that ϕ is equivalent with Φ^n_ϕ. In particular, any sentence of SCL which expresses a property that is not FO*-definable is undetermined in some model.*

Proof.

If ϕ is determined everywhere, then $\phi \vee \neg\phi$ is valid. This in turn implies that $\Phi^n_{\phi\vee\neg\phi} = \Phi^n_\phi \vee \Phi^n_{\neg\phi}$ is also valid, for some $n \in \mathbb{N}$. We claim that ϕ is equivalent with Φ^n_ϕ. First, we have that $\Phi^n_\phi \models \phi$. Secondly, since $\Phi^n_\phi \vee \Phi^n_{\neg\phi}$ is valid, we have that $\neg\Phi^n_\phi \models \Phi^n_{\neg\phi} \models \neg\phi$. □

- If we know that ϕ is determined everywhere, then we can effectively find an approximant Φ^n_ϕ which is equivalent with ϕ.

Everywhere determined sentences: the finite case

Theorem

Restrict attention to finite linearly ordered structures.

Everywhere determined sentences: the finite case

Theorem

Restrict attention to finite linearly ordered structures. For every sentence ϕ of SCL there exists a sentence ϕ' of SCL such that

Everywhere determined sentences: the finite case

Theorem

Restrict attention to finite linearly ordered structures. For every sentence ϕ of SCL there exists a sentence ϕ' of SCL such that

1. *for every structure $\mathfrak{A}$ we have that*

$$\mathfrak{A} \models \phi \Leftrightarrow \mathfrak{A} \models \phi'$$

and

Everywhere determined sentences: the finite case

Theorem

Restrict attention to finite linearly ordered structures. For every sentence ϕ of SCL there exists a sentence ϕ' of SCL such that

1. *for every structure $\mathfrak{A}$ we have that*

$$\mathfrak{A} \models \phi \Leftrightarrow \mathfrak{A} \models \phi'$$

   and

2. *ϕ' is determined in every structure.*

Main open problems

1. Are the satisfiability problems of BndSCL^2 and SCL^2 decidable?
2. Is BndSCL contained in SCL?

Main open problems

1. Are the satisfiability problems of BndSCL^2 and SCL^2 decidable?
2. Is BndSCL contained in SCL?

Thanks! :-)