Kleene Star in Substructural Logics

Stepan Kuznetsov

WDCM 2022 · Novosibirsk & Kazan / online

Steklov Mathematical Institute, RAS

- This talk is about **infinitary extensions** of certain **propositional non-classical logics.**

Substructural Logics

- This talk is about **infinitary extensions** of certain **propositional non-classical logics.**
- The logics in question are **substructural logics,** which get their name due to the fact that they lack all or some of the **structural rules:**

$$\frac{\Gamma, \Delta \to C}{\Gamma, A, \Delta \to C}\ W \qquad \frac{\Gamma, A, A, \Delta \to C}{\Gamma, A, \Delta \to C}\ C \qquad \frac{\Gamma, A, B, \Delta \to C}{\Gamma, B, A, \Delta \to C}\ P$$

Substructural Logics

- This talk is about **infinitary extensions** of certain **propositional non-classical logics.**
- The logics in question are **substructural logics,** which get their name due to the fact that they lack all or some of the **structural rules:**

$$\frac{\Gamma, \Delta \to C}{\Gamma, A, \Delta \to C}\ W \qquad \frac{\Gamma, A, A, \Delta \to C}{\Gamma, A, \Delta \to C}\ C \qquad \frac{\Gamma, A, B, \Delta \to C}{\Gamma, B, A, \Delta \to C}\ P$$

- These rules are called weakening, contraction, and permutation (exchange).

Substructural Logics

- This talk is about **infinitary extensions** of certain **propositional non-classical logics.**
- The logics in question are **substructural logics,** which get their name due to the fact that they lack all or some of the **structural rules:**

$$\frac{\Gamma, \Delta \to C}{\Gamma, A, \Delta \to C}\ W \qquad \frac{\Gamma, A, A, \Delta \to C}{\Gamma, A, \Delta \to C}\ C \qquad \frac{\Gamma, A, B, \Delta \to C}{\Gamma, B, A, \Delta \to C}\ P$$

- These rules are called weakening, contraction, and permutation (exchange).
- Here they are formulated as Gentzen-style sequent rules, but they can also be written algebraically: $A \to 1$; $A \to A \otimes A$; $A \otimes B = B \otimes A$ (here $\otimes$ is a sort of conjunction and 1 is constant "true").

Substructural Logics: Motivations

- Substructurality arises from various sources.

Substructural Logics: Motivations

- Substructurality arises from various sources.
- One motivation, going back to Girard's (1987) **linear logic,** is connected to considering formulae as **resources.**

Substructural Logics: Motivations

- Substructurality arises from various sources.
- One motivation, going back to Girard's (1987) **linear logic,** is connected to considering formulae as **resources.**
- That informal interpretation invalidates contraction (a resource may not be used twice) and weakening (resources should not be wasted).

Substructural Logics: Motivations

- Substructurality arises from various sources.
- One motivation, going back to Girard's (1987) **linear logic,** is connected to considering formulae as **resources.**
- That informal interpretation invalidates contraction (a resource may not be used twice) and weakening (resources should not be wasted).
- However, permutation is still valid, as resources may be used in arbitrary order.

Substructural Logics: Motivations

- Substructurality arises from various sources.
- One motivation, going back to Girard's (1987) **linear logic,** is connected to considering formulae as **resources.**
- That informal interpretation invalidates contraction (a resource may not be used twice) and weakening (resources should not be wasted).
- However, permutation is still valid, as resources may be used in arbitrary order.
- The most well-known **non-commutative** substructural logic is the **Lambek calculus** (1958), introduced for mathematical description of natural languages.

Substructural Logics: Motivations

- Substructurality arises from various sources.
- One motivation, going back to Girard's (1987) **linear logic,** is connected to considering formulae as **resources.**
- That informal interpretation invalidates contraction (a resource may not be used twice) and weakening (resources should not be wasted).
- However, permutation is still valid, as resources may be used in arbitrary order.
- The most well-known **non-commutative** substructural logic is the **Lambek calculus** (1958), introduced for mathematical description of natural languages.
- Linguistic "resources," i.e., syntactic structures, are in general non-commutative, since the word order matters (at least in English).

Substructural Logics: Motivation

- For us, the motivation is complexity.

Substructural Logics: Motivation

- For us, the motivation is complexity.
- We shall consider extensions of substructural logics with one certain fixpoint operator, the **Kleene star.**

Substructural Logics: Motivation

- For us, the motivation is complexity.
- We shall consider extensions of substructural logics with one certain fixpoint operator, the **Kleene star.**
- We shall mostly be interested in **infinitary** systems with Kleene star, by means of ω-rules.

Substructural Logics: Motivation

- For us, the motivation is complexity.
- We shall consider extensions of substructural logics with one certain fixpoint operator, the **Kleene star.**
- We shall mostly be interested in **infinitary** systems with Kleene star, by means of ω-rules.
- Such systems are usually not r.e., and have various complexity levels, ranging from Π_1^0 to Π_1^1.

Substructural Logics: Motivation

- For us, the motivation is complexity.
- We shall consider extensions of substructural logics with one certain fixpoint operator, the **Kleene star.**
- We shall mostly be interested in **infinitary** systems with Kleene star, by means of ω-rules.
- Such systems are usually not r.e., and have various complexity levels, ranging from Π^0_1 to Π^1_1.
- Another interesting complexity measure is the **closure ordinal,** which we shall define later.

Substructural Logics: Motivation

- For us, the motivation is complexity.
- We shall consider extensions of substructural logics with one certain fixpoint operator, the **Kleene star.**
- We shall mostly be interested in **infinitary** systems with Kleene star, by means of ω-rules.
- Such systems are usually not r.e., and have various complexity levels, ranging from Π^0_1 to Π^1_1.
- Another interesting complexity measure is the **closure ordinal,** which we shall define later.
- In general, here we have a quite simple, purely propositional framework, in which we could obtain very high and interesting complexity bounds.

MALC: the Basic Substructural Logic

- The basic (associative) substructural logic is the **multiplicative-additive Lambek calculus MALC** (Lambek 1958, van Benthem 1991, Kanazawa 1992).

MALC: the Basic Substructural Logic

- The basic (associative) substructural logic is the **multiplicative-additive Lambek calculus MALC** (Lambek 1958, van Benthem 1991, Kanazawa 1992).
- Its axioms and rules are as follows:

$$\frac{}{A \to A}\ Id \qquad \frac{}{\Gamma, 0, \Delta \to B}\ 0L \qquad \frac{\Gamma, \Delta \to B}{\Gamma, 1, \Delta \to B}\ 1L \qquad \frac{}{\to 1}\ 1R$$

$$\frac{\Pi \to A \quad \Gamma, B, \Delta \to C}{\Gamma, \Pi, A \multimap B, \Delta \to C}\ {\multimap}L \qquad \frac{A, \Pi \to B}{\Pi \to A \multimap B}\ {\multimap}R \qquad \frac{\Gamma, A, B, \Delta \to B}{\Gamma, A \otimes B, \Delta \to B}\ {\otimes}L$$

$$\frac{\Pi \to A \quad \Gamma, B, \Delta \to C}{\Gamma, B \multimapinv A, \Pi, \Delta \to C}\ {\multimapinv}L \qquad \frac{\Pi, A \to B}{\Pi \to B \multimapinv A}\ {\multimapinv}R \qquad \frac{\Gamma \to A \quad \Delta \to B}{\Gamma, \Delta \to A \otimes B}\ {\otimes}R$$

$$\frac{\Gamma, A, \Delta \to C \quad \Gamma, B, \Delta \to C}{\Gamma, A \vee B, \Delta \to C}\ {\vee}L \qquad \frac{\Pi \to A}{\Pi \to A \vee B} \quad \frac{\Pi \to B}{\Pi \to A \vee B}\ {\vee}R$$

$$\frac{\Gamma, A, \Delta \to C}{\Gamma, A \wedge B, \Delta \to C} \quad \frac{\Gamma, B, \Delta \to C}{\Gamma, A \wedge B, \Delta \to C}\ {\wedge}L \qquad \frac{\Pi \to A \quad \Pi \to B}{\Pi \to A \wedge B}\ {\wedge}R$$

Cut

- The Cut rule of the form

$$\frac{\Pi \to A \quad \Gamma, A, \Delta \to C}{\Gamma, \Pi, \Delta \to C}\ \mathit{Cut}$$

  is admissible in **MALC**.

Cut

- The Cut rule of the form

$$\frac{\Pi \to A \quad \Gamma, A, \Delta \to C}{\Gamma, \Pi, \Delta \to C}\ \mathit{Cut}$$

  is admissible in **MALC**.
- This is proved by a standard induction; even easier, than for intuitionistic or classical propositional logic.

Cut

- The Cut rule of the form

$$\frac{\Pi \to A \quad \Gamma, A, \Delta \to C}{\Gamma, \Pi, \Delta \to C}\ Cut$$

  is admissible in **MALC**.
- This is proved by a standard induction; even easier, than for intuitionistic or classical propositional logic.
- The cut elimination argument can be further propagated (as a transfinite one) to infinitary extensions of **MALC**.

Algebraic Semantics for MALC

- The natural class of algebraic models for **MALC** is formed by **residuated lattices** (see Galatos et al. 2007).

Algebraic Semantics for MALC

- The natural class of algebraic models for **MALC** is formed by **residuated lattices** (see Galatos et al. 2007).
- A residuated lattice combines the lattice structure: $\wedge$, $\vee$, 0, and $\preceq$ for $\rightarrow$, along with the structure of a monoid: $\otimes$, 1.

Algebraic Semantics for MALC

- The natural class of algebraic models for **MALC** is formed by **residuated lattices** (see Galatos et al. 2007).
- A residuated lattice combines the lattice structure: $\wedge$, $\vee$, 0, and $\preceq$ for $\to$, along with the structure of a monoid: $\otimes$, 1.
- $\multimap$ and $\circ\!-$ are **residuals** of $\otimes$ w.r.t. $\preceq$:

$$A \preceq C \circ\!- B \iff A \otimes B \preceq C \iff B \preceq A \multimap B.$$

Algebraic Semantics for MALC

- The natural class of algebraic models for **MALC** is formed by **residuated lattices** (see Galatos et al. 2007).
- A residuated lattice combines the lattice structure: $\wedge$, $\vee$, 0, and $\preceq$ for $\rightarrow$, along with the structure of a monoid: $\otimes$, 1.
- $\multimap$ and $\circ\!-$ are **residuals** of $\otimes$ w.r.t. $\preceq$:

$$A \preceq C \circ\!- B \iff A \otimes B \preceq C \iff B \preceq A \multimap B.$$

- Notice that identifying $\otimes$ and $\wedge$ (two "conjunctions") makes a residuated lattice a Heyting algebra.

Algebraic Semantics for MALC

- The natural class of algebraic models for **MALC** is formed by **residuated lattices** (see Galatos et al. 2007).
- A residuated lattice combines the lattice structure: $\wedge$, $\vee$, 0, and $\preceq$ for $\rightarrow$, along with the structure of a monoid: $\otimes$, 1.
- $\multimap$ and $\multimapinv$ are **residuals** of $\otimes$ w.r.t. $\preceq$:

$$A \preceq C \multimapinv B \iff A \otimes B \preceq C \iff B \preceq A \multimap B.$$

- Notice that identifying $\otimes$ and $\wedge$ (two "conjunctions") makes a residuated lattice a Heyting algebra.
- In other words, this restores all structural rules.

Algebraic Semantics for MALC

- The natural class of algebraic models for **MALC** is formed by **residuated lattices** (see Galatos et al. 2007).
- A residuated lattice combines the lattice structure: $\wedge$, $\vee$, 0, and $\preceq$ for $\to$, along with the structure of a monoid: $\otimes$, 1.
- $\multimap$ and $\multimapinv$ are **residuals** of $\otimes$ w.r.t. $\preceq$:

$$A \preceq C \multimapinv B \iff A \otimes B \preceq C \iff B \preceq A \multimap B.$$

- Notice that identifying $\otimes$ and $\wedge$ (two "conjunctions") makes a residuated lattice a Heyting algebra.
- In other words, this restores all structural rules.
- There are, however, more interesting examples of residuated lattices.

L-models and R-models

- We shall consider two classes of residuated lattices, in which the lattice part is set-theoretic.

L-models and R-models

- We shall consider two classes of residuated lattices, in which the lattice part is set-theoretic.
- **L-models** are interpretations of **MALC** on residuated lattices of **formal languages.**

L-models and R-models

- We shall consider two classes of residuated lattices, in which the lattice part is set-theoretic.
- **L-models** are interpretations of **MALC** on residuated lattices of **formal languages.**
- In L-models, multiplication and residuals are defined as follows:

$$A \otimes B = \{uv \mid u \in A, v \in B\}$$

$$A \multimap B = \{v \mid (\forall u \in A)\, uv \in B\}$$

$$B \multimapinv A = \{u \mid (\forall v \in A)\, uv \in B\}$$

L-models and R-models

- We shall consider two classes of residuated lattices, in which the lattice part is set-theoretic.
- **L-models** are interpretations of **MALC** on residuated lattices of **formal languages.**
- In L-models, multiplication and residuals are defined as follows:

$$A \otimes B = \{uv \mid u \in A, v \in B\}$$
$$A \multimap B = \{v \mid (\forall u \in A)\, uv \in B\}$$
$$B \multimapinv A = \{u \mid (\forall v \in A)\, uv \in B\}$$

- This corresponds to Lambek's original linguistic ideas.

L-models and R-models

- **R-models** are interpretations of **MALC** on residuated lattices of **binary relations** (on a set W), with the following multiplication and residuals:

$$A \otimes B = A \circ B = \{(x, z) \mid (\exists y \in W)\,((x, y) \in A, (y, z) \in B)\}$$

$$A \multimap B = \{(y, z) \mid (\forall x \in W)\,((x, y) \in A \Rightarrow (x, z) \in B)\}$$

$$B \multimapinv A = \{(x, y) \mid (\forall z \in W)\,((y, z) \in A \Rightarrow (x, z) \in B)\}$$

L-models and R-models

- **R-models** are interpretations of **MALC** on residuated lattices of **binary relations** (on a set W), with the following multiplication and residuals:

$$A \otimes B = A \circ B = \{(x,z) \mid (\exists y \in W)\,((x,y) \in A, (y,z) \in B)\}$$

$$A \multimap B = \{(y,z) \mid (\forall x \in W)\,((x,y) \in A \Rightarrow (x,z) \in B)\}$$

$$B \multimapinv A = \{(x,y) \mid (\forall z \in W)\,((y,z) \in A \Rightarrow (x,z) \in B)\}$$

- Relations may be viewed as (non-deterministic) **actions** in a transition system; residuals are **conditional** actions.

L-models and R-models

- **R-models** are interpretations of **MALC** on residuated lattices of **binary relations** (on a set W), with the following multiplication and residuals:

$$A \otimes B = A \circ B = \{(x,z) \mid (\exists y \in W)\,((x,y) \in A, (y,z) \in B)\}$$

$$A \multimap B = \{(y,z) \mid (\forall x \in W)\,((x,y) \in A \Rightarrow (x,z) \in B)\}$$

$$B \multimapinv A = \{(x,y) \mid (\forall z \in W)\,((y,z) \in A \Rightarrow (x,z) \in B)\}$$

- Relations may be viewed as (non-deterministic) **actions** in a transition system; residuals are **conditional** actions.
- **MALC** is of course sound w.r.t. both L- and R-models. Completeness is a subtle issue, we shall not discuss it now.

Kleene Star

- Both L-models and R-models allow adding **iteration** or **Kleene star,** as a unary operation.

Kleene Star

- Both L-models and R-models allow adding **iteration** or **Kleene star,** as a unary operation.
- In L-models, we have

$$A^* = \bigcup_{n=0}^{\infty} A^n = \{u_1 \dots u_n \mid n \in \mathbb{N},\ u_1, \dots, u_n \in A\}.$$

Kleene Star

- Both L-models and R-models allow adding **iteration** or **Kleene star,** as a unary operation.
- In L-models, we have

$$A^* = \bigcup_{n=0}^{\infty} A^n = \{u_1 \ldots u_n \mid n \in \mathbb{N},\ u_1, \ldots, u_n \in A\}.$$

- In R-models, A^* is the reflexive-transitive closure of relation A.

Kleene Star

- Both L-models and R-models allow adding **iteration** or **Kleene star,** as a unary operation.
- In L-models, we have

$$A^* = \bigcup_{n=0}^{\infty} A^n = \{u_1 \dots u_n \mid n \in \mathbb{N},\ u_1, \dots, u_n \in A\}.$$

- In R-models, A^* is the reflexive-transitive closure of relation A.
- This definition of Kleene star as an infinite join is called the ***-continuous** definition.

Kleene Star

- Both L-models and R-models allow adding **iteration** or **Kleene star,** as a unary operation.
- In L-models, we have

$$A^* = \bigcup_{n=0}^{\infty} A^n = \{u_1 \dots u_n \mid n \in \mathbb{N},\ u_1, \dots, u_n \in A\}.$$

- In R-models, A^* is the reflexive-transitive closure of relation A.
- This definition of Kleene star as an infinite join is called the ***-continuous** definition.
- At the same time, A^* is the least fixpoint of $X \mapsto 1 \vee (A \otimes X)$.

Kleene Star

- Both L-models and R-models allow adding **iteration** or **Kleene star,** as a unary operation.
- In L-models, we have

$$A^* = \bigcup_{n=0}^{\infty} A^n = \{u_1 \dots u_n \mid n \in \mathbb{N},\ u_1, \dots, u_n \in A\}.$$

- In R-models, A^* is the reflexive-transitive closure of relation A.
- This definition of Kleene star as an infinite join is called the ***-continuous** definition.
- At the same time, A^* is the least fixpoint of $X \mapsto 1 \vee (A \otimes X)$.
- In our specific models, these two definitions give the same result.

Kleene Star

- In general, this is not the case.

Kleene Star

- In general, this is not the case.
- Residuated lattices, extended with Kleene star defined as the fixpoint, are called **action lattices** (Pratt 1991, Kozen 1994) or **residuated Kleene lattices.**

Kleene Star

- In general, this is not the case.
- Residuated lattices, extended with Kleene star defined as the fixpoint, are called **action lattices** (Pratt 1991, Kozen 1994) or **residuated Kleene lattices.**
- The *-continuous definition of Kleene star gives a narrower class of ***-continuous action lattices.**

Kleene Star

- In general, this is not the case.
- Residuated lattices, extended with Kleene star defined as the fixpoint, are called **action lattices** (Pratt 1991, Kozen 1994) or **residuated Kleene lattices.**
- The *-continuous definition of Kleene star gives a narrower class of ***-continuous action lattices.**
- Non-*-continuous action lattices exist, although such examples are artificial.

(Infinitary) Action Logic

- The two definitions of Kleene star give rise to two axiomatizations for this operations, both extending **MALC**.

(Infinitary) Action Logic

- The two definitions of Kleene star give rise to two axiomatizations for this operations, both extending **MALC**.
- For the *-continuous definition, we obtain **infinitary action logic $\mathbf{ACT}_\omega$** (Palka 2007):

$$\frac{\left(\Gamma, A^n, \Delta \to C\right)_{n=0}^{\infty}}{\Gamma, A^*, \Delta \to C}\ {*L_\omega} \qquad \frac{\Pi_1 \to A \quad \ldots \quad \Pi_n \to A}{\Pi_1, \ldots, \Pi_n \to A^*}\ {*R_n},\ n \geq 0$$

(Infinitary) Action Logic

- The two definitions of Kleene star give rise to two axiomatizations for this operations, both extending **MALC**.
- For the *-continuous definition, we obtain **infinitary action logic ACT$_\omega$** (Palka 2007):

$$\frac{\left(\Gamma, A^n, \Delta \to C\right)_{n=0}^{\infty}}{\Gamma, A^*, \Delta \to C}\ {*L_\omega} \qquad \frac{\Pi_1 \to A \quad \ldots \quad \Pi_n \to A}{\Pi_1, \ldots, \Pi_n \to A^*}\ {*R_n},\ n \geq 0$$

- The fixpoint definition yields **action logic ACT** (Kozen 1994):

$$\frac{\to B \quad A, B \to B}{A^* \to B} \qquad \frac{}{\to A^*} \qquad \frac{\Gamma \to A \quad \Delta \to A^*}{\Gamma, \Delta \to A^*}$$

$$\frac{\Pi \to A \quad \Gamma, A, \Delta \to C}{\Gamma, \Pi, \Delta \to C}\ Cut$$

(Infinitary) Action Logic

- The two definitions of Kleene star give rise to two axiomatizations for this operations, both extending **MALC**.
- For the *-continuous definition, we obtain **infinitary action logic ACT**$_\omega$ (Palka 2007):

$$\frac{\left(\Gamma, A^n, \Delta \to C\right)_{n=0}^{\infty}}{\Gamma, A^*, \Delta \to C}\ {*L_\omega} \qquad \frac{\Pi_1 \to A \quad \dots \quad \Pi_n \to A}{\Pi_1, \dots, \Pi_n \to A^*}\ {*R_n},\ n \geq 0$$

- The fixpoint definition yields **action logic ACT** (Kozen 1994):

$$\frac{\to B \quad A, B \to B}{A^* \to B} \qquad \frac{}{\to A^*} \qquad \frac{\Gamma \to A \quad \Delta \to A^*}{\Gamma, \Delta \to A^*}$$

$$\frac{\Pi \to A \quad \Gamma, A, \Delta \to C}{\Gamma, \Pi, \Delta \to C}\ Cut$$

- Cut is eliminable in **ACT**$_\omega$ (Palka 2007); for **ACT**, constructing a cut-free system is an open question.

Complexity of Action Logic

Theorem (Buszkowski & Palka 2007)

$\mathbf{ACT}_\omega$ *is* Π^0_1*-complete.*

Theorem (K. 2019–20)

$\mathbf{ACT}$ *is* Σ^0_1*-complete.*

Complexity of Action Logic

Theorem (Buszkowski & Palka 2007)

$\mathbf{ACT}_\omega$ *is* Π^0_1*-complete.*

Theorem (K. 2019–20)

$\mathbf{ACT}$ *is* Σ^0_1*-complete.*

- In particular, **ACT** has strictly less theorems than $\mathbf{ACT}_\omega$.

Complexity of Action Logic

Theorem (Buszkowski & Palka 2007)

$\mathbf{ACT}_\omega$ *is* Π_1^0*-complete.*

Theorem (K. 2019–20)

$\mathbf{ACT}$ *is* Σ_1^0*-complete.*

- In particular, **ACT** has strictly less theorems than $\mathbf{ACT}_\omega$.
- $\mathbf{ACT}_\omega$ is not r.e., thus, it is not axiomatizable by finite means; the usage of infinitary (somewhat model-theoretic) mechanisms like ω-rules is inevitable.

Complexity of Action Logic

Theorem (Buszkowski & Palka 2007)

$\mathbf{ACT}_\omega$ *is* Π^0_1*-complete.*

Theorem (K. 2019–20)

$\mathbf{ACT}$ *is* Σ^0_1*-complete.*

- In particular, **ACT** has strictly less theorems than $\mathbf{ACT}_\omega$.
- $\mathbf{ACT}_\omega$ is not r.e., thus, it is not axiomatizable by finite means; the usage of infinitary (somewhat model-theoretic) mechanisms like ω-rules is inevitable.
- On the other hand, $\mathbf{ACT}_\omega$ (unlike **ACT**) enjoys the finite model property (Buszkowski & Palka 2007).

Commutative Action Logic

- In what follows, we shift to the commutative situation and consider $\mathbf{CommACT}_\omega$ and $\mathbf{CommACT}$.

Commutative Action Logic

- In what follows, we shift to the commutative situation and consider $\mathbf{CommACT}_\omega$ and **CommACT**.
- This can be obtained by adding permutation (structural rule):

$$\frac{\Gamma, A, B, \Delta \to C}{\Gamma, B, A, \Delta \to C}\ P$$

Commutative Action Logic

- In what follows, we shift to the commutative situation and consider $\mathbf{CommACT}_\omega$ and **CommACT**.
- This can be obtained by adding permutation (structural rule):

$$\frac{\Gamma, A, B, \Delta \to C}{\Gamma, B, A, \Delta \to C}\ P$$

- In the commutative case, we have only one division operation: $A \multimap B \equiv B \circ\!\!- A$.

Commutative Action Logic

- In what follows, we shift to the commutative situation and consider $\mathbf{CommACT}_\omega$ and $\mathbf{CommACT}$.
- This can be obtained by adding permutation (structural rule):

$$\frac{\Gamma, A, B, \Delta \to C}{\Gamma, B, A, \Delta \to C}\ P$$

- In the commutative case, we have only one division operation: $A \multimap B \equiv B \circ\!\!- A$.
- Our natural models are non-commutative, but the proofs in the commutative case are simpler and easier to follow.

Commutative Action Logic

- In what follows, we shift to the commutative situation and consider $\mathbf{CommACT}_\omega$ and $\mathbf{CommACT}$.
- This can be obtained by adding permutation (structural rule):

$$\frac{\Gamma, A, B, \Delta \to C}{\Gamma, B, A, \Delta \to C}\ P$$

- In the commutative case, we have only one division operation: $A \multimap B \equiv B \multimapinv A$.
- Our natural models are non-commutative, but the proofs in the commutative case are simpler and easier to follow.

Theorem (K. 2020–22)

$\mathbf{CommACT}_\omega$ *is* Π^0_1*-complete.* $\mathbf{CommACT}$ *is* Σ^0_1*-complete.*

Minsky Machines

- A Minsky machine $\mathscr{M}$ operates several *counters* (registers), which hold natural numbers.

Minsky Machines

- A Minsky machine $\mathscr{M}$ operates several *counters* (registers), which hold natural numbers.
- Instructions of $\mathscr{M}$ can be of the following sorts:

$\textsc{inc}(p, r, q)$	being in state p, increase register r by 1 and move to state q;
$\textsc{jzdec}(p, r, q_0, q_1)$	being in state p, check whether the value of r is 0: if yes, move to state q_0, if no, decrease r by 1 and move to state q_1.

Minsky Machines

- A Minsky machine $\mathscr{M}$ operates several *counters* (registers), which hold natural numbers.
- Instructions of $\mathscr{M}$ can be of the following sorts:

$\textsc{inc}(p, r, q)$	being in state p, increase register r by 1 and move to state q;
$\textsc{jzdec}(p, r, q_0, q_1)$	being in state p, check whether the value of r is 0: if yes, move to state q_0, if no, decrease r by 1 and move to state q_1.

- We consider only *deterministic* Minsky machines.

Minsky Machines

- A Minsky machine $\mathscr{M}$ operates several *counters* (registers), which hold natural numbers.
- Instructions of $\mathscr{M}$ can be of the following sorts:

$\textsc{inc}(p, r, q)$	being in state p, increase register r by 1 and move to state q;
$\textsc{jzdec}(p, r, q_0, q_1)$	being in state p, check whether the value of r is 0: if yes, move to state q_0, if no, decrease r by 1 and move to state q_1.

- We consider only *deterministic* Minsky machines.
- Two counters are sufficient for a Σ_1^0-complete halting problem (Minsky 1961).

Minsky Machines

- A Minsky machine $\mathscr{M}$ operates several *counters* (registers), which hold natural numbers.
- Instructions of $\mathscr{M}$ can be of the following sorts:

$\textsc{inc}(p, r, q)$	being in state p, increase register r by 1 and move to state q;
$\textsc{jzdec}(p, r, q_0, q_1)$	being in state p, check whether the value of r is 0: if yes, move to state q_0, if no, decrease r by 1 and move to state q_1.

- We consider only *deterministic* Minsky machines.
- Two counters are sufficient for a Σ^0_1-complete halting problem (Minsky 1961).
- ... thus, *non-halting* is Π^0_1-complete.

Minsky Machines

- A Minsky machine $\mathscr{M}$ operates several *counters* (registers), which hold natural numbers.
- Instructions of $\mathscr{M}$ can be of the following sorts:

$\textsc{inc}(p, r, q)$	being in state p, increase register r by 1 and move to state q;
$\textsc{jzdec}(p, r, q_0, q_1)$	being in state p, check whether the value of r is 0: if yes, move to state q_0, if no, decrease r by 1 and move to state q_1.

- We consider only *deterministic* Minsky machines.
- Two counters are sufficient for a Σ^0_1-complete halting problem (Minsky 1961).
- ... thus, *non-halting* is Π^0_1-complete.
- Sometimes it is more convenient to use three counters.

Encoding Minsky Instructions

- Each instruction I of M is encoded by a formula A_I:

$$
\begin{aligned}
A_{\textsc{inc}(p,r,q)} &= p \multimap (q \otimes r) \\
A_{\textsc{jzdec}(p,r,q_0,q_1)} &= ((p \otimes r) \multimap q_1) \wedge (p \multimap (q_0 \vee z_r)).
\end{aligned}
$$

Encoding Minsky Instructions

- Each instruction I of M is encoded by a formula A_I:

$$A_{\textsc{inc}(p,r,q)} = p \multimap (q \otimes r)$$
$$A_{\textsc{jzdec}(p,r,q_0,q_1)} = ((p \otimes r) \multimap q_1) \wedge (p \multimap (q_0 \vee z_r)).$$

- Moreover, we add three extra formulae: $N_r = z_r \multimap z_r$ for each counter r (i.e., a, b, or c).

Encoding Minsky Instructions

- Each instruction I of M is encoded by a formula A_I:

$$A_{\textsc{inc}(p,r,q)} = p \multimap (q \otimes r)$$
$$A_{\textsc{jzdec}(p,r,q_0,q_1)} = ((p \otimes r) \multimap q_1) \wedge (p \multimap (q_0 \vee z_r)).$$

- Moreover, we add three extra formulae: $N_r = z_r \multimap z_r$ for each counter r (i.e., a, b, or c).
- The encoding is due to Lincoln et al. 1992.

Encoding Minsky Instructions

- Each instruction I of M is encoded by a formula A_I:

$$A_{\textsc{inc}(p,r,q)} = p \multimap (q \otimes r)$$
$$A_{\textsc{jzdec}(p,r,q_0,q_1)} = ((p \otimes r) \multimap q_1) \wedge (p \multimap (q_0 \vee z_r)).$$

- Moreover, we add three extra formulae: $N_r = z_r \multimap z_r$ for each counter r (i.e., a, b, or c).
- The encoding is due to Lincoln et al. 1992.
- However, we now consider *non-halting* instead of halting, and model it using Kleene star instead of exponential.

Encoding Minsky Instructions

- Each instruction I of M is encoded by a formula A_I:

$$A_{\textsc{inc}(p,r,q)} = p \multimap (q \otimes r)$$
$$A_{\textsc{jzdec}(p,r,q_0,q_1)} = ((p \otimes r) \multimap q_1) \wedge (p \multimap (q_0 \vee z_r)).$$

- Moreover, we add three extra formulae: $N_r = z_r \multimap z_r$ for each counter r (i.e., a, b, or c).
- The encoding is due to Lincoln et al. 1992.
- However, we now consider *non-halting* instead of halting, and model it using Kleene star instead of exponential.
- Also, in succedents of our sequents we now have to represent an *arbitrary* configuration of the Minsky machine being encoded, which is also implemented using Kleene star.

Encoding Infinite Execution

$$E = \bigwedge_{I} A_I \wedge N_{\mathrm{a}} \wedge N_{\mathrm{b}} \wedge N_{\mathrm{c}}$$

Encoding Infinite Execution

$E = \bigwedge_{I} A_I \wedge N_{\mathrm{a}} \wedge N_{\mathrm{b}} \wedge N_{\mathrm{c}}$ ← this formula encodes the machine

Encoding Infinite Execution

$$E = \bigwedge_I A_I \wedge N_{\mathrm{a}} \wedge N_{\mathrm{b}} \wedge N_{\mathrm{c}}$$ ← this formula encodes the machine

$$D = \left(\mathrm{a}^* \otimes \mathrm{b}^* \otimes \mathrm{c}^* \otimes \bigvee_{q \in Q} q\right) \vee (\mathrm{b}^* \otimes \mathrm{c}^* \otimes z_{\mathrm{a}}) \vee (\mathrm{a}^* \otimes \mathrm{c}^* \otimes z_{\mathrm{b}}) \vee (\mathrm{a}^* \otimes \mathrm{b}^* \otimes z_{\mathrm{c}})$$

Encoding Infinite Execution

$$E = \bigwedge_I A_I \wedge N_{\mathrm{a}} \wedge N_{\mathrm{b}} \wedge N_{\mathrm{c}}$$ ← this formula encodes the machine

$$D = \Big(\mathrm{a}^* \otimes \mathrm{b}^* \otimes \mathrm{c}^* \otimes \bigvee_{q \in Q} q\Big) \vee (\mathrm{b}^* \otimes \mathrm{c}^* \otimes z_{\mathrm{a}}) \vee (\mathrm{a}^* \otimes \mathrm{c}^* \otimes z_{\mathrm{b}}) \vee (\mathrm{a}^* \otimes \mathrm{b}^* \otimes z_{\mathrm{c}})$$

↑

this formula encodes any valid configuration

Encoding Infinite Execution

$$E = \bigwedge_I A_I \wedge N_\mathrm{a} \wedge N_\mathrm{b} \wedge N_\mathrm{c}$$ ← this formula encodes the machine

$$D = \left(\mathrm{a}^* \otimes \mathrm{b}^* \otimes \mathrm{c}^* \otimes \bigvee_{q \in Q} q\right) \vee (\mathrm{b}^* \otimes \mathrm{c}^* \otimes z_\mathrm{a}) \vee (\mathrm{a}^* \otimes \mathrm{c}^* \otimes z_\mathrm{b}) \vee (\mathrm{a}^* \otimes \mathrm{b}^* \otimes z_\mathrm{c})$$

↑

this formula encodes any valid configuration

z_a, z_b, z_c are for zero check in JZDEC, run in parallel with the main execution.

Encoding Infinite Execution

$$E = \bigwedge_I A_I \wedge N_{\mathrm{a}} \wedge N_{\mathrm{b}} \wedge N_{\mathrm{c}}$$ ← this formula encodes the machine

$$D = \left(\mathrm{a}^* \otimes \mathrm{b}^* \otimes \mathrm{c}^* \otimes \bigvee_{q \in Q} q\right) \vee (\mathrm{b}^* \otimes \mathrm{c}^* \otimes z_{\mathrm{a}}) \vee (\mathrm{a}^* \otimes \mathrm{c}^* \otimes z_{\mathrm{b}}) \vee (\mathrm{a}^* \otimes \mathrm{b}^* \otimes z_{\mathrm{c}})$$

↑

this formula encodes any valid configuration

$z_{\mathrm{a}}, z_{\mathrm{b}}, z_{\mathrm{c}}$ are for zero check in JZDEC, run in parallel with the main execution.

Lemma

$E^*, \mathrm{a}^a, \mathrm{b}^b, \mathrm{c}^c, q \to D$ *is derivable iff the machine runs infinitely starting from* (q, a, b, c).

Encoding Infinite Computation

- $E^*, \mathrm{a}^a, \mathrm{b}^b, \mathrm{c}^c, q \to D$ is derivable if and only if so is $E^n, \mathrm{a}^a, \mathrm{b}^b, \mathrm{c}^c, q \to D$ for any $n \geq 0$.

Encoding Infinite Computation

- $E^*, \mathrm{a}^a, \mathrm{b}^b, \mathrm{c}^c, q \to D$ is derivable if and only if so is $E^n, \mathrm{a}^a, \mathrm{b}^b, \mathrm{c}^c, q \to D$ for any $n \geq 0$.
- This corresponds to n steps of execution.

Encoding Infinite Computation

- $E^*, \mathrm{a}^a, \mathrm{b}^b, \mathrm{c}^c, q \to D$ is derivable if and only if so is $E^n, \mathrm{a}^a, \mathrm{b}^b, \mathrm{c}^c, q \to D$ for any $n \geq 0$.
- This corresponds to n steps of execution.
- Since our machine is deterministic, partial computations form an infinite one. (In the non-deterministic case, use Kőnig's lemma.)

Encoding Infinite Computation

- $E^*, \mathrm{a}^a, \mathrm{b}^b, \mathrm{c}^c, q \to D$ is derivable if and only if so is $E^n, \mathrm{a}^a, \mathrm{b}^b, \mathrm{c}^c, q \to D$ for any $n \geq 0$.
- This corresponds to n steps of execution.
- Since our machine is deterministic, partial computations form an infinite one. (In the non-deterministic case, use Kőnig's lemma.)
- Base case: $n = 0$, and $\mathrm{a}^a, \mathrm{b}^b, \mathrm{b}^c, q \to D$ is derivable ((q, a, b, c) is a valid configuration).

Encoding $\textsc{inc}(p, a, q)$

$$
\dfrac{\dfrac{p \to p \qquad \dfrac{E^{k-1}, \mathrm{a}^{a+1}, \mathrm{b}^{b}, \mathrm{c}^{c}, q \to D}{E^{k-1}, \mathrm{a}^{a}, \mathrm{b}^{b}, \mathrm{c}^{c}, q \otimes \mathrm{a} \to D}\ \otimes L}{E^{k-1}, A_{\textsc{inc}(p,\mathrm{a},q)}, \mathrm{a}^{a}, \mathrm{b}^{b}, \mathrm{c}^{c}, p \to D}\ \multimap L\ (A_{\textsc{inc}(p,\mathrm{a},q)} = p \multimap (q \otimes \mathrm{a}))}{E^{k}, \mathrm{a}^{a}, \mathrm{b}^{b}, \mathrm{c}^{c}, p \to D}\ \wedge L \text{ several times}
$$

Encoding $\textsc{jzdec}(p, a, q)$

- $a \neq 0$

$$
\dfrac{\dfrac{p \to p \quad \mathsf{a} \to \mathsf{a}}{p, \mathsf{a} \to p \otimes \mathsf{a}}\ \otimes R \qquad \color{red}{E^{k-1}}\color{black}{, \mathsf{a}^{a-1}, \mathsf{b}^{b}, \mathsf{c}^{c}, q_1 \to} \color{blue}{D}}{\dfrac{\dfrac{\color{red}{E^{k-1}, (p \otimes \mathsf{a}) \multimap q_1}\color{black}{, \mathsf{a}^{a}, \mathsf{b}^{b}, \mathsf{c}^{c}, p \to} \color{blue}{D}}{\color{red}{E^{k-1}, A_{\textsc{jzdec}(p,\mathsf{a},q_0,q_1)}}\color{black}{, \mathsf{a}^{a}, \mathsf{b}^{b}, \mathsf{c}^{c}, p \to} \color{blue}{D}}\ \wedge L}{\color{red}{E^{k}}\color{black}{, \mathsf{a}^{a}, \mathsf{b}^{b}, \mathsf{c}^{c}, p \to} \color{blue}{D}}\ \wedge L \text{ several times}}\ \multimap L
$$

Encoding $\textsc{jzdec}(p, a, q)$

- $a \neq 0$

$$\dfrac{\dfrac{\dfrac{\dfrac{p \to p \quad \mathrm{a} \to \mathrm{a}}{p, \mathrm{a} \to p \otimes \mathrm{a}}\otimes R \qquad E^{k-1}, \mathrm{a}^{a-1}, \mathrm{b}^{b}, \mathrm{c}^{c}, q_1 \to D}{E^{k-1}, (p \otimes \mathrm{a}) \multimap q_1, \mathrm{a}^{a}, \mathrm{b}^{b}, \mathrm{c}^{c}, p \to D}\multimap L}{E^{k-1}, A_{\textsc{jzdec}(p,\mathrm{a},q_0,q_1)}, \mathrm{a}^{a}, \mathrm{b}^{b}, \mathrm{c}^{c}, p \to D}\wedge L}{E^{k}, \mathrm{a}^{a}, \mathrm{b}^{b}, \mathrm{c}^{c}, p \to D}\wedge L \text{ several times}$$

- $a = 0$

$$\dfrac{\dfrac{\dfrac{p \to p \qquad \dfrac{E^{k-1}, \mathrm{b}^{b}, \mathrm{c}^{c}, q_0 \to D \qquad \dfrac{(z_{\mathrm{a}} \multimap z_{\mathrm{a}})^{k-1}, \mathrm{b}^{b}, \mathrm{c}^{c}, z_{\mathrm{a}} \to D}{E^{k-1}, \mathrm{b}^{b}, \mathrm{c}^{c}, z_{\mathrm{a}} \to D}\wedge L \text{ s.t.}}{E^{k-1}, q_0 \vee z_{\mathrm{a}}, \mathrm{b}^{b}, \mathrm{c}^{c} \to D}\vee L}{E^{k-1}, p \multimap (q_0 \vee z_{\mathrm{a}}), \mathrm{b}^{b}, \mathrm{c}^{c}, p \to D}\multimap L}{E^{k-1}, A_{\textsc{jzdec}(p,\mathrm{a},q_0,q_1)}, \mathrm{b}^{b}, \mathrm{c}^{c}, p \to D}\wedge L}{E^{k}, \mathrm{b}^{b}, p \to D}\wedge L \text{ s.t.}$$

Lemma

If the machine runs **circularly** *starting from* (q, a, b, c)*, then* $E^*, \mathrm{a}^a, \mathrm{b}^b, \mathrm{c}^c, q \to D$ *admits a circular proof, thus, a proof in* **CommACT**.

$$
\dfrac{q \to D \qquad \dfrac{\dfrac{\dfrac{p, \mathrm{a}^a, \mathrm{b}^b, \mathrm{c}^c \to D \qquad \dfrac{\dfrac{E^*, p, \mathrm{a}^a, \mathrm{b}^b, \mathrm{c}^c \to D}{\vdots}}{E^*, E, p, \mathrm{a}^a, \mathrm{b}^b, \mathrm{c}^c \to D}}{E^*, p, \mathrm{a}^a, \mathrm{b}^b, \mathrm{c}^c \to D}\ {*L}}{\vdots}}{E^*, E, q \to D}}{E^*, q \to D}\ {*L}
$$

Inseparability

- The converse, however, is not true. For example, if the machine just increases one counter, then $E^*, \mathrm{a}^a, \mathrm{b}^b, \mathrm{c}^c, q \to D$ is also derivable in **CommACT**.

Inseparability

- The converse, however, is not true. For example, if the machine just increases one counter, then $E^*, \mathrm{a}^a, \mathrm{b}^b, \mathrm{c}^c, q \to D$ is also derivable in **CommACT**.
- Therefore, we use an indirect technique for proving complexity, based on **effective inseparability.**

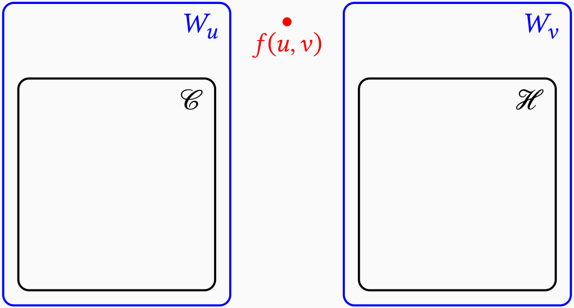
Inseparability

- The converse, however, is not true. For example, if the machine just increases one counter, then $E^*, \mathrm{a}^a, \mathrm{b}^b, \mathrm{c}^c, q \to D$ is also derivable in **CommACT**.
- Therefore, we use an indirect technique for proving complexity, based on **effective inseparability.**
- Let $\mathscr{C}$ be the set of machines and input data such that the machine works circularly, and let $\mathscr{H}$ be the one where the machine halts.

Inseparability

- The converse, however, is not true. For example, if the machine just increases one counter, then $E^*, \mathrm{a}^a, \mathrm{b}^b, \mathrm{c}^c, q \to D$ is also derivable in **CommACT**.
- Therefore, we use an indirect technique for proving complexity, based on **effective inseparability.**
- Let $\mathscr{C}$ be the set of machines and input data such that the machine works circularly, and let $\mathscr{H}$ be the one where the machine halts.
- **Folklore:** $\mathscr{C}$ and $\mathscr{H}$ are effectively inseparable.

Effective Inseparability



If one tries to separate $\mathscr{C}$ from $\mathscr{H}$ by a pair of disjoint r.e. sets W_u and W_v (where W_x is "the x-th r.e. set"), aiming make a decidable separator $W_u = \overline{W}_v$, the other player can falsify this separation via a computable function f.

Effective Inseparability and Σ_1^0-Completeness

- It follows from a result by Myhill (1955) that if A separates two effectively inseparable sets (e.g., $\mathscr{C}$ and $\mathscr{H}$) and A is itself r.e., then A is Σ_1^0-complete.

Effective Inseparability and Σ^0_1-Completeness

- It follows from a result by Myhill (1955) that if A separates two effectively inseparable sets (e.g., $\mathscr{C}$ and $\mathscr{H}$) and A is itself r.e., then A is Σ^0_1-complete.
- Therefore, we obtain Σ^0_1-completeness of **CommACT**.

Effective Inseparability and Σ_1^0-Completeness

- It follows from a result by Myhill (1955) that if A separates two effectively inseparable sets (e.g., $\mathscr{C}$ and $\mathscr{H}$) and A is itself r.e., then A is Σ_1^0-complete.
- Therefore, we obtain Σ_1^0-completeness of **CommACT**.
- The reasoning in the non-commutative case is similar, however, the encoding of computations is more involved.

Effective Inseparability and Σ^0_1-Completeness

- It follows from a result by Myhill (1955) that if A separates two effectively inseparable sets (e.g., $\mathscr{C}$ and $\mathscr{H}$) and A is itself r.e., then A is Σ^0_1-complete.
- Therefore, we obtain Σ^0_1-completeness of **CommACT**.
- The reasoning in the non-commutative case is similar, however, the encoding of computations is more involved.
- This non-commutative encoding goes via the totality problem for context-free grammars (Buszkowski 2007, K. 2019).

Effective Inseparability and Σ^0_1-Completeness

- It follows from a result by Myhill (1955) that if A separates two effectively inseparable sets (e.g., $\mathscr{C}$ and $\mathscr{H}$) and A is itself r.e., then A is Σ^0_1-complete.
- Therefore, we obtain Σ^0_1-completeness of **CommACT**.
- The reasoning in the non-commutative case is similar, however, the encoding of computations is more involved.
- This non-commutative encoding goes via the totality problem for context-free grammars (Buszkowski 2007, K. 2019).
- A direct encoding would not work, since there is no way of moving formulae around.

Upper Bounds and Closure Ordinals

- The Π^0_1 upper bound for infinitary (commutative) action logic can be proved by Palka's ***-elimination** method.

Upper Bounds and Closure Ordinals

- The Π^0_1 upper bound for infinitary (commutative) action logic can be proved by Palka's ***-elimination** method.
- The idea is to replace each negative occurrence of A^* with its "n-th approximation" $1 \vee A \vee A^2 \vee \ldots \vee A^n$, and then show that a sequent is derivable iff all its approximations ("$\forall n$" quantifier) are.

Upper Bounds and Closure Ordinals

- The Π^0_1 upper bound for infinitary (commutative) action logic can be proved by Palka's ***-elimination** method.
- The idea is to replace each negative occurrence of A^* with its "n-th approximation" $1 \vee A \vee A^2 \vee \ldots \vee A^n$, and then show that a sequent is derivable iff all its approximations ("$\forall n$" quantifier) are.
- Another proof of Π^0_1-boundedness was given by Das & Pous (2017) in terms of non-well-founded derivations.

Upper Bounds and Closure Ordinals

- The Π^0_1 upper bound for infinitary (commutative) action logic can be proved by Palka's ***-elimination** method.
- The idea is to replace each negative occurrence of A^* with its "n-th approximation" $1 \vee A \vee A^2 \vee \ldots \vee A^n$, and then show that a sequent is derivable iff all its approximations ("$\forall n$" quantifier) are.
- Another proof of Π^0_1-boundedness was given by Das & Pous (2017) in terms of non-well-founded derivations.
- We shall show yet another, more robust approach, based on **closure ordinals.**

Closure Ordinal

- Let $\mathscr{D} : \mathscr{P}(\mathrm{Seq}) \to \mathscr{P}(\mathrm{Seq})$ denote the **immediate derivability** operator on sequents, for the given calculus.

Closure Ordinal

- Let $\mathscr{D} : \mathscr{P}(\mathrm{Seq}) \to \mathscr{P}(\mathrm{Seq})$ denote the **immediate derivability** operator on sequents, for the given calculus.
- By default we consider cut-free derivability.

Closure Ordinal

- Let $\mathscr{D} : \mathscr{P}(\mathrm{Seq}) \to \mathscr{P}(\mathrm{Seq})$ denote the **immediate derivability** operator on sequents, for the given calculus.
- By default we consider cut-free derivability.
- Let $S_\alpha = \mathscr{D}^\alpha(\emptyset)$, where $\alpha \in \mathrm{Ord}$, form the transfinite sequence of iterations of $\mathscr{D}$.

Closure Ordinal

- Let $\mathscr{D} : \mathscr{P}(\mathrm{Seq}) \to \mathscr{P}(\mathrm{Seq})$ denote the **immediate derivability** operator on sequents, for the given calculus.
- By default we consider cut-free derivability.
- Let $S_\alpha = \mathscr{D}^\alpha(\emptyset)$, where $\alpha \in \mathrm{Ord}$, form the transfinite sequence of iterations of $\mathscr{D}$.
- By Knaster − Tarski, this sequence has a fixed point.

Closure Ordinal

- Let $\mathscr{D} : \mathscr{P}(\mathrm{Seq}) \to \mathscr{P}(\mathrm{Seq})$ denote the **immediate derivability** operator on sequents, for the given calculus.
- By default we consider cut-free derivability.
- Let $S_\alpha = \mathscr{D}^\alpha(\emptyset)$, where $\alpha \in \mathrm{Ord}$, form the transfinite sequence of iterations of $\mathscr{D}$.
- By Knaster − Tarski, this sequence has a fixed point.
- The smallest ordinal α_0 at which the sequence stabilizes ($S_{\alpha_0} = S_{\alpha_0+1} = \ldots$) is called the **closure ordinal** for $\mathscr{D}$; the set S_{α_0} is exactly the set of all derivable sequents.

Closure Ordinal

- Let $\mathscr{D} : \mathscr{P}(\mathrm{Seq}) \to \mathscr{P}(\mathrm{Seq})$ denote the **immediate derivability** operator on sequents, for the given calculus.
- By default we consider cut-free derivability.
- Let $S_\alpha = \mathscr{D}^\alpha(\varnothing)$, where $\alpha \in \mathrm{Ord}$, form the transfinite sequence of iterations of $\mathscr{D}$.
- By Knaster − Tarski, this sequence has a fixed point.
- The smallest ordinal α_0 at which the sequence stabilizes ($S_{\alpha_0} = S_{\alpha_0+1} = \ldots$) is called the **closure ordinal** for $\mathscr{D}$; the set S_{α_0} is exactly the set of all derivable sequents.
- The closure ordinal is yet another complexity measure, along with the undecidability degree.

Closure Ordinal

- Let $\mathscr{D} : \mathscr{P}(\mathrm{Seq}) \to \mathscr{P}(\mathrm{Seq})$ denote the **immediate derivability** operator on sequents, for the given calculus.
- By default we consider cut-free derivability.
- Let $S_\alpha = \mathscr{D}^\alpha(\varnothing)$, where $\alpha \in \mathrm{Ord}$, form the transfinite sequence of iterations of $\mathscr{D}$.
- By Knaster − Tarski, this sequence has a fixed point.
- The smallest ordinal α_0 at which the sequence stabilizes ($S_{\alpha_0} = S_{\alpha_0+1} = \ldots$) is called the **closure ordinal** for $\mathscr{D}$; the set S_{α_0} is exactly the set of all derivable sequents.
- The closure ordinal is yet another complexity measure, along with the undecidability degree.
- For finitary calculi, like **ACT**, the closure ordinal is trivially ω, so we now consider only infinitary calculi.

Closure Ordinal

- A general folklore result: for any monotone Π^1_1 operator $F : \mathscr{P}(\mathbb{N}) \to \mathscr{P}(\mathbb{N})$ its least fixed point is Π^1_1-bounded and its closure ordinal is $\leq \omega_1^{\mathrm{CK}}$ (Church–Kleene ordinal, the first non-constructive one).

Closure Ordinal

- A general folklore result: for any monotone Π^1_1 operator $F : \mathscr{P}(\mathbb{N}) \to \mathscr{P}(\mathbb{N})$ its least fixed point is Π^1_1-bounded and its closure ordinal is $\leq \omega_1^{\mathrm{CK}}$ (Church–Kleene ordinal, the first non-constructive one).
- This applies to our $\mathscr{D}$, which yields $\alpha_0 \leq \omega_1^{\mathrm{CK}}$.

Closure Ordinal

- A general folklore result: for any monotone Π^1_1 operator $F: \mathscr{P}(\mathbb{N}) \to \mathscr{P}(\mathbb{N})$ its least fixed point is Π^1_1-bounded and its closure ordinal is $\leq \omega_1^{\mathrm{CK}}$ (Church–Kleene ordinal, the first non-constructive one).
- This applies to our $\mathscr{D}$, which yields $\alpha_0 \leq \omega_1^{\mathrm{CK}}$.
- However, for $\mathbf{ACT}_\omega$ a better bound can be obtained.

Closure Ordinal

- A general folklore result: for any monotone Π^1_1 operator $F : \mathscr{P}(\mathbb{N}) \to \mathscr{P}(\mathbb{N})$ its least fixed point is Π^1_1-bounded and its closure ordinal is $\leq \omega_1^{\mathrm{CK}}$ (Church–Kleene ordinal, the first non-constructive one).
- This applies to our $\mathscr{D}$, which yields $\alpha_0 \leq \omega_1^{\mathrm{CK}}$.
- However, for $\mathbf{ACT}_\omega$ a better bound can be obtained.
- Namely, there exists an ordinal measure **(rank)** on sequents, $\eta : \mathrm{Seq} \to \omega^\omega$, such that each rule strictly increases this measure.

Rank and Complexity

- For each formula or sequent, η yields a polynomial of ω (a.k.a. Cantor normal form)

$$\omega^n c_n + \ldots + \omega c_1 + c_0,\ c_i \in \mathbb{N}.$$

Rank and Complexity

- For each formula or sequent, η yields a polynomial of ω (a.k.a. Cantor normal form)

$$\omega^n c_n + \ldots + \omega c_1 + c_0,\ c_i \in \mathbb{N}.$$

- $\eta(A^*) = \omega \cdot \eta(A)$, and for binary operations we take componentwise sum plus 1 ($c_i = c_i' + c_i''$ for $i > 0$ and $c_0 = c_0' + c_0'' + 1$).

Rank and Complexity

- For each formula or sequent, η yields a polynomial of ω (a.k.a. Cantor normal form)

$$\omega^n c_n + \ldots + \omega c_1 + c_0,\ c_i \in \mathbb{N}.$$

- $\eta(A^*) = \omega \cdot \eta(A)$, and for binary operations we take componentwise sum plus 1 ($c_i = c_i' + c_i''$ for $i > 0$ and $c_0 = c_0' + c_0'' + 1$).
- Each rule strictly increases η. Hence, if $\Pi \to C$ is derivable and $\alpha = \eta(\Pi \to C)$, then $(\Pi \to C) \in S_\alpha$.

Rank and Complexity

- For each formula or sequent, η yields a polynomial of ω (a.k.a. Cantor normal form)

$$\omega^n c_n + \ldots + \omega c_1 + c_0,\ c_i \in \mathbb{N}.$$

- $\eta(A^*) = \omega \cdot \eta(A)$, and for binary operations we take componentwise sum plus 1 ($c_i = c_i' + c_i''$ for $i > 0$ and $c_0 = c_0' + c_0'' + 1$).
- Each rule strictly increases η. Hence, if $\Pi \to C$ is derivable and $\alpha = \eta(\Pi \to C)$, then $(\Pi \to C) \in S_\alpha$.
- Therefore, $\alpha_0 \leq \omega^\omega$.

Rank and Complexity

- For each formula or sequent, η yields a polynomial of ω (a.k.a. Cantor normal form)

$$\omega^n c_n + \ldots + \omega c_1 + c_0,\ c_i \in \mathbb{N}.$$

- $\eta(A^*) = \omega \cdot \eta(A)$, and for binary operations we take componentwise sum plus 1 ($c_i = c_i' + c_i''$ for $i > 0$ and $c_0 = c_0' + c_0'' + 1$).
- Each rule strictly increases η. Hence, if $\Pi \to C$ is derivable and $\alpha = \eta(\Pi \to C)$, then $(\Pi \to C) \in S_\alpha$.
- Therefore, $\alpha_0 \leq \omega^\omega$.
- **Conjecture:** the closure ordinal for $\mathbf{ACT}_\omega$ is *exactly* ω^ω.

Rank and Complexity

- The ω^ω bound on the closure ordinal is used in yet another proof of the Π^0_1 upper bound for $\mathbf{ACT}_\omega$ (K. & Speranski 2022).

Rank and Complexity

- The ω^ω bound on the closure ordinal is used in yet another proof of the Π^0_1 upper bound for $\mathbf{ACT}_\omega$ (K. & Speranski 2022).
- Namely, we prove that the set of (Gödel numbers of) sequents of rank α derivable in $\mathbf{ACT}_\omega$ is Π^0_1.

Rank and Complexity

- The ω^ω bound on the closure ordinal is used in yet another proof of the Π^0_1 upper bound for $\mathbf{ACT}_\omega$ (K. & Speranski 2022).
- Namely, we prove that the set of (Gödel numbers of) sequents of rank α derivable in $\mathbf{ACT}_\omega$ is Π^0_1.
- Moreover, the Π^0_1 formula defining this set is uniform, i.e., computable from α by a computable function g.

Rank and Complexity

- The ω^ω bound on the closure ordinal is used in yet another proof of the Π^0_1 upper bound for $\mathbf{ACT}_\omega$ (K. & Speranski 2022).
- Namely, we prove that the set of (Gödel numbers of) sequents of rank α derivable in $\mathbf{ACT}_\omega$ is Π^0_1.
- Moreover, the Π^0_1 formula defining this set is uniform, i.e., computable from α by a computable function g.
- Then we define the set of all derivable sequents uniformly using Π^0_1-SAT, which is also Π^0_1:

$$\Pi^0_1\text{-SAT}(x, g(\dot{\eta}(x))).$$

Rank and Complexity

- For this argument to work, it is crucial that for each sequent there is a **finite** number of possible immediate derivations.

Rank and Complexity

- For this argument to work, it is crucial that for each sequent there is a **finite** number of possible immediate derivations.
- Informally, here we iterate the arithmetical universal (Π^0_1) quantifier, corresponding to $*L_\omega$, at most ω^ω times, interleaving with decidable proof search by other rules.

Rank and Complexity

- For this argument to work, it is crucial that for each sequent there is a **finite** number of possible immediate derivations.
- Informally, here we iterate the arithmetical universal (Π^0_1) quantifier, corresponding to $*L_\omega$, at most ω^ω times, interleaving with decidable proof search by other rules.
- This can be "compressed" into one Π^0_1.

Rank and Complexity

- For this argument to work, it is crucial that for each sequent there is a **finite** number of possible immediate derivations.
- Informally, here we iterate the arithmetical universal (Π^0_1) quantifier, corresponding to $*L_\omega$, at most ω^ω times, interleaving with decidable proof search by other rules.
- This can be "compressed" into one Π^0_1.
- The same argument works for $\mathbf{CommACT}_\omega$.

(Sub)exponentials

- Now let us extend $\mathbf{ACT}_\omega$ with the **exponential** modality taken from linear logic (Girard 1987).

(Sub)exponentials

- Now let us extend $\mathbf{ACT}_\omega$ with the **exponential** modality taken from linear logic (Girard 1987).
- This modality locally allows structural rules (permutation, contraction, weakening), which are absent in the original system.

(Sub)exponentials

- Now let us extend $\mathbf{ACT}_\omega$ with the **exponential** modality taken from linear logic (Girard 1987).
- This modality locally allows structural rules (permutation, contraction, weakening), which are absent in the original system.
- For simplicity, we consider only one exponential modality, denoted by $!A$, but the arguments are easily extendable to polymodal systems with **subexponentials** (Danos et al. 1993, Kanovich et al. 2019).

(Sub)exponentials

- Now let us extend $\mathbf{ACT}_\omega$ with the **exponential** modality taken from linear logic (Girard 1987).
- This modality locally allows structural rules (permutation, contraction, weakening), which are absent in the original system.
- For simplicity, we consider only one exponential modality, denoted by $!A$, but the arguments are easily extendable to polymodal systems with **subexponentials** (Danos et al. 1993, Kanovich et al. 2019).
- (Sub)exponential extensions are considered both for $\mathbf{ACT}_\omega$ and for $\mathbf{CommACT}_\omega$.

Exponential

- The rules for ! are as follows:

$$\frac{\Gamma, A, \Delta \to C}{\Gamma, !A, \Delta \to C}\ !L \qquad \frac{!A_1, \dots, !A_n \to B}{!A_1, \dots, !A_n \to !B}\ !R$$

$$\frac{\Gamma, !A, !A, \Delta \to C}{\Gamma, !A, \Delta \to C}\ !C \qquad \frac{\Gamma, \Delta \to C}{\Gamma, !A, \Delta \to C}\ !W$$

$$\frac{\Gamma, \Pi, !A, \Delta \to C}{\Gamma, !A, \Pi, \Delta \to C} \quad \frac{\Gamma, !A, \Pi, \Delta \to C}{\Gamma, \Pi, !A, \Delta \to C}\ !P$$

Exponential

- The rules for ! are as follows:

$$\frac{\Gamma, A, \Delta \to C}{\Gamma, !A, \Delta \to C}\ !L \qquad \frac{!A_1, \ldots, !A_n \to B}{!A_1, \ldots, !A_n \to !B}\ !R$$

$$\frac{\Gamma, !A, !A, \Delta \to C}{\Gamma, !A, \Delta \to C}\ !C \qquad \frac{\Gamma, \Delta \to C}{\Gamma, !A, \Delta \to C}\ !W$$

$$\frac{\Gamma, \Pi, !A, \Delta \to C}{\Gamma, !A, \Pi, \Delta \to C} \quad \frac{\Gamma, !A, \Pi, \Delta \to C}{\Gamma, \Pi, !A, \Delta \to C}\ !P$$

- The extension of **MALC** with the exponential (without Kleene star) is Σ_1^0-complete (Lincoln et al. 1992).

Exponential

- The rules for ! are as follows:

$$\frac{\Gamma, A, \Delta \to C}{\Gamma, !A, \Delta \to C}\ !L \qquad \frac{!A_1, \ldots, !A_n \to B}{!A_1, \ldots, !A_n \to !B}\ !R$$

$$\frac{\Gamma, !A, !A, \Delta \to C}{\Gamma, !A, \Delta \to C}\ !C \qquad \frac{\Gamma, \Delta \to C}{\Gamma, !A, \Delta \to C}\ !W$$

$$\frac{\Gamma, \Pi, !A, \Delta \to C}{\Gamma, !A, \Pi, \Delta \to C} \quad \frac{\Gamma, !A, \Pi, \Delta \to C}{\Gamma, \Pi, !A, \Delta \to C}\ !P$$

- The extension of **MALC** with the exponential (without Kleene star) is Σ_1^0-complete (Lincoln et al. 1992).
- We show that for the corresponding extension of $\mathbf{ACT}_\omega$ (denoted by $\mathbf{!ACT}_\omega$) complexity raises dramatically.

Complexity of $!\mathbf{ACT}_\omega$

Theorem (K. & Speranski 2022)

$!\mathbf{ACT}_\omega$ *is* Π^1_1*-complete, and its closure ordinal is* ω_1^{CK}.

Complexity of !ACT$_\omega$

Theorem (K. & Speranski 2022)

!ACT$_\omega$ *is* Π^1_1*-complete, and its closure ordinal is* ω_1^{CK}.

- The upper bounds follow from the general folklore result, thanks to the fact that our $\mathscr{D}$ operator is a monotone Π^1_1 one.

Complexity of $!\mathbf{ACT}_\omega$

Theorem (K. & Speranski 2022)

$!\mathbf{ACT}_\omega$ is Π^1_1-complete, and its closure ordinal is ω_1^{CK}.

- The upper bounds follow from the general folklore result, thanks to the fact that our $\mathscr{D}$ operator is a monotone Π^1_1 one.
- For the lower complexity bound, Π^1_1-hardness, we use the exponential modality to encode derivability from finite sets of hypotheses (non-logical axioms).

Complexity of $!\mathbf{ACT}_\omega$

Theorem (K. & Speranski 2022)

$!\mathbf{ACT}_\omega$ is Π^1_1-complete, and its closure ordinal is ω_1^{CK}.

- The upper bounds follow from the general folklore result, thanks to the fact that our $\mathscr{D}$ operator is a monotone Π^1_1 one.
- For the lower complexity bound, Π^1_1-hardness, we use the exponential modality to encode derivability from finite sets of hypotheses (non-logical axioms).
- The latter is Π^1_1-complete due to Kozen (2002), by encoding of well-foundedness of recursively defined graphs on $\mathbb{N}$.

Complexity of !ACT$_\omega$

Theorem (K. & Speranski 2022)

!ACT$_\omega$ *is* Π^1_1*-complete, and its closure ordinal is* ω_1^{CK}.

- The upper bounds follow from the general folklore result, thanks to the fact that our $\mathscr{D}$ operator is a monotone Π^1_1 one.
- For the lower complexity bound, Π^1_1-hardness, we use the exponential modality to encode derivability from finite sets of hypotheses (non-logical axioms).
- The latter is Π^1_1-complete due to Kozen (2002), by encoding of well-foundedness of recursively defined graphs on $\mathbb{N}$.
- We conjecture that a variant of Kozen's argument works in the commutative setting, thus yielding the same complexity results for **!CommACT**$_\omega$.

Complexity of !ACT$_\omega$

- As for the closure ordinal of **!ACT**$_\omega$, we prove that if it were $< \omega_1^{\mathrm{CK}}$, then the derivability problem would be hyperarithmetical, thus not Π_1^1-hard.

Complexity of $!\mathbf{ACT}_\omega$

- As for the closure ordinal of $!\mathbf{ACT}_\omega$, we prove that if it were $< \omega_1^{\mathrm{CK}}$, then the derivability problem would be hyperarithmetical, thus not Π^1_1-hard.
- We make use of the fact that our $\mathscr{D}$ operator is not only Π^1_1, but also Σ^1_1 (in fact, it is much simpler).

Complexity of $!\mathbf{ACT}_\omega$

- As for the closure ordinal of $!\mathbf{ACT}_\omega$, we prove that if it were $< \omega_1^{\mathrm{CK}}$, then the derivability problem would be hyperarithmetical, thus not Π_1^1-hard.
- We make use of the fact that our $\mathscr{D}$ operator is not only Π_1^1, but also Σ_1^1 (in fact, it is much simpler).
- For Kleene's $\mathscr{O}$ notation for ω_1^{CK}, denoted by $\nu_{\mathscr{O}}$, we have a computable function which maps n to a Σ_1^1-formula defining $S_{\nu_{\mathscr{O}}(n)}$.

Complexity of $!\mathbf{ACT}_\omega$

- As for the closure ordinal of $!\mathbf{ACT}_\omega$, we prove that if it were $< \omega_1^{\mathrm{CK}}$, then the derivability problem would be hyperarithmetical, thus not Π_1^1-hard.
- We make use of the fact that our $\mathscr{D}$ operator is not only Π_1^1, but also Σ_1^1 (in fact, it is much simpler).
- For Kleene's $\mathcal{O}$ notation for ω_1^{CK}, denoted by $\nu_{\mathcal{O}}$, we have a computable function which maps n to a Σ_1^1-formula defining $S_{\nu_{\mathcal{O}}(n)}$.
- In particular, if the closure ordinal is a constructive ordinal $\alpha_0 = \nu_{\mathcal{O}}(n_0)$, we show that S_{α_0} belongs to Σ_1^1.

Complexity of $!\mathbf{ACT}_\omega$

- As for the closure ordinal of $!\mathbf{ACT}_\omega$, we prove that if it were $< \omega_1^{\mathrm{CK}}$, then the derivability problem would be hyperarithmetical, thus not Π_1^1-hard.
- We make use of the fact that our $\mathscr{D}$ operator is not only Π_1^1, but also Σ_1^1 (in fact, it is much simpler).
- For Kleene's $\mathscr{O}$ notation for ω_1^{CK}, denoted by $\nu_{\mathscr{O}}$, we have a computable function which maps n to a Σ_1^1-formula defining $S_{\nu_{\mathscr{O}}(n)}$.
- In particular, if the closure ordinal is a constructive ordinal $\alpha_0 = \nu_{\mathscr{O}}(n_0)$, we show that S_{α_0} belongs to Σ_1^1.
- Therefore, it is in $\Pi_1^1 \cap \Sigma_1^1 = \Delta_1^1$, i.e., it is hyperarithmetical.

Multiplexing

- An intermediate complexity can be obtained with a subexponential allowing the **multiplexing** rule:

$$\frac{\Gamma, \overbrace{A, \ldots, A}^{n}, \Delta \to C}{\Gamma, !^{\mathsf{m}} A, \Delta \to C} \; !^{\mathsf{m}} M,\; n \geqslant 0 \qquad \frac{A \to B}{!^{\mathsf{m}} A \to !^{\mathsf{m}} B} \; !^{\mathsf{m}} R$$

Multiplexing

- An intermediate complexity can be obtained with a subexponential allowing the **multiplexing** rule:

$$\frac{\Gamma, \overbrace{A, \dots, A}^{n}, \Delta \to C}{\Gamma, !^{\mathsf{m}} A, \Delta \to C}\ !^{\mathsf{m}} M,\ n \geqslant 0 \qquad \frac{A \to B}{!^{\mathsf{m}} A \to !^{\mathsf{m}} B}\ !^{\mathsf{m}} R$$

- These are the only rules for ! in the commutative situation; in the non-commutative case, we add another modality for local commutativity:

$$\frac{\Gamma, A, \Delta \to C}{\Gamma, \nabla A, \Delta \to C}\ \nabla L \qquad \frac{A \to B}{\nabla A \to \nabla B}\ \nabla R \qquad \frac{\Gamma, B, \nabla A, \Delta \to B}{\Gamma, \nabla A, B, \Delta \to C} \quad \frac{\Gamma, \nabla A, B, \Delta \to C}{\Gamma, B, \nabla A, \Delta \to B}\ \nabla P$$

Multiplexing

- An intermediate complexity can be obtained with a subexponential allowing the **multiplexing** rule:

$$\frac{\Gamma, \overbrace{A, \ldots, A}^{n}, \Delta \to C}{\Gamma, !^{\mathrm{m}} A, \Delta \to C}\ !^{\mathrm{m}}M,\ n \geqslant 0 \qquad \frac{A \to B}{!^{\mathrm{m}} A \to !^{\mathrm{m}} B}\ !^{\mathrm{m}}R$$

- These are the only rules for ! in the commutative situation; in the non-commutative case, we add another modality for local commutativity:

$$\frac{\Gamma, A, \Delta \to C}{\Gamma, \nabla A, \Delta \to C}\ \nabla L \qquad \frac{A \to B}{\nabla A \to \nabla B}\ \nabla R \qquad \frac{\Gamma, B, \nabla A, \Delta \to B}{\Gamma, \nabla A, B, \Delta \to C} \quad \frac{\Gamma, \nabla A, B, \Delta \to C}{\Gamma, B, \nabla A, \Delta \to B}\ \nabla P$$

- **MALC** extended with $!^{\mathrm{m}}$ and ∇ is Σ_1^0-complete (Kanovich et al. 2020).

Multiplexing

- An intermediate complexity can be obtained with a subexponential allowing the **multiplexing** rule:

$$\frac{\Gamma, \overbrace{A, \ldots, A}^{n}, \Delta \to C}{\Gamma, !^{\mathrm{m}} A, \Delta \to C}\ !^{\mathrm{m}} M,\ n \geqslant 0 \qquad \frac{A \to B}{!^{\mathrm{m}} A \to !^{\mathrm{m}} B}\ !^{\mathrm{m}} R$$

- These are the only rules for ! in the commutative situation; in the non-commutative case, we add another modality for local commutativity:

$$\frac{\Gamma, A, \Delta \to C}{\Gamma, \nabla A, \Delta \to C}\ \nabla L \qquad \frac{A \to B}{\nabla A \to \nabla B}\ \nabla R \qquad \frac{\Gamma, B, \nabla A, \Delta \to B}{\Gamma, \nabla A, B, \Delta \to C} \quad \frac{\Gamma, \nabla A, B, \Delta \to C}{\Gamma, B, \nabla A, \Delta \to B}\ \nabla P$$

- **MALC** extended with $!^{\mathrm{m}}$ and ∇ is Σ_1^0-complete (Kanovich et al. 2020).
- In the infinitary situation, however, multiplexing is significantly weaker than contraction.

Multiplexing

- Namely, we can use the same ranking function as for $\mathbf{ACT}_\omega$, extending it with $\eta(!^{\mathsf{m}} A) = \omega \cdot \eta(A)$.

Multiplexing

- Namely, we can use the same ranking function as for $\mathbf{ACT}_\omega$, extending it with $\eta(!^{\mathsf{m}} A) = \omega \cdot \eta(A)$.
- This makes the closure ordinal bounded by ω^ω, and we can pinpoint an upper bound for complexity of decision problem in the hyperarithmetical hierarchy:

Multiplexing

- Namely, we can use the same ranking function as for $\mathbf{ACT}_\omega$, extending it with $\eta(!^{\mathsf{m}} A) = \omega \cdot \eta(A)$.
- This makes the closure ordinal bounded by ω^ω, and we can pinpoint an upper bound for complexity of decision problem in the hyperarithmetical hierarchy:

Theorem (K. & Speranski 2022)

Decidability in $!^{\mathsf{m}}\nabla\mathbf{ACT}_\omega$ *belongs to* $\Sigma^0_{\omega^\omega}$, *and the closure ordinal is* $\leq \omega^\omega$. *The same for* $!^{\mathsf{m}}\mathbf{CommACT}_\omega$.

Multiplexing

- Namely, we can use the same ranking function as for $\mathbf{ACT}_\omega$, extending it with $\eta(!^{\mathsf{m}} A) = \omega \cdot \eta(A)$.
- This makes the closure ordinal bounded by ω^ω, and we can pinpoint an upper bound for complexity of decision problem in the hyperarithmetical hierarchy:

Theorem (K. & Speranski 2022)

Decidability in $!^{\mathsf{m}}\nabla\mathbf{ACT}_\omega$ *belongs to* $\Sigma^0_{\omega^\omega}$, *and the closure ordinal is* $\leq \omega^\omega$. *The same for* $!^{\mathsf{m}}\mathbf{CommACT}_\omega$.

- Informally, we iterate the ω-rule and "enumerable" proof search with finite rules, at most ω^ω times.

Multiplexing: Lower Bound

- We prove arithmetical lower bound for infinitary action logic with multiplexing.

Multiplexing: Lower Bound

- We prove arithmetical lower bound for infinitary action logic with multiplexing.

Theorem

$!^{\mathrm{m}}\nabla\mathbf{ACT}_\omega$ and $!^{\mathrm{m}}\mathbf{CommACT}_\omega$ are Π^0_{2k+1}-hard for any $k \in \omega$.

Multiplexing: Lower Bound

- We prove arithmetical lower bound for infinitary action logic with multiplexing.

Theorem

$!^{\mathrm{m}}\nabla\mathbf{ACT}_\omega$ and $!^{\mathrm{m}}\mathbf{CommACT}_\omega$ are Π^0_{2k+1}-hard for any $k \in \omega$.

- For $!^{\mathrm{m}}\mathbf{CommACT}_\omega$ we encode the following problem:

$$\forall n_1 > 0\ \exists n_2 > 0\ \forall n_3 > 0\ \ldots\ \exists n_{2k} > 0$$

$$\big(\text{machine } \mathscr{M} \text{ does not halt when started from } (q_0; n_1, \ldots, n_{2k}, 0, 0, 0)\big)$$

Multiplexing: Lower Bound

- We prove arithmetical lower bound for infinitary action logic with multiplexing.

Theorem

$!^{\mathrm{m}}\nabla\mathbf{ACT}_\omega$ *and* $!^{\mathrm{m}}\mathbf{CommACT}_\omega$ *are* Π^0_{2k+1}*-hard for any* $k \in \omega$.

- For $!^{\mathrm{m}}\mathbf{CommACT}_\omega$ we encode the following problem:

$$\forall n_1 > 0\ \exists n_2 > 0\ \forall n_3 > 0\ \dots\ \exists n_{2k} > 0$$

$$\big(\text{machine } \mathscr{M} \text{ does not halt when started from } (q_0; n_1, \dots, n_{2k}, 0, 0, 0)\big)$$

- This is encoded using the “key-and-lock” technique.

Multiplexing: Lower Bound

- The quantifier is encoded as follows:

$$K_m = \begin{cases} p_{m-1} \multimap (a_m^+ \cdot p_m) & \text{if } m \text{ is odd;} \\ p_{m-1} \multimap !(a_m \cdot !p_m) & \text{if } m \text{ is even.} \end{cases}$$

Multiplexing: Lower Bound

- The quantifier is encoded as follows:

$$K_m = \begin{cases} p_{m-1} \multimap (a_m^+ \cdot p_m) & \text{if } m \text{ is odd;} \\ p_{m-1} \multimap !(a_m \cdot !p_m) & \text{if } m \text{ is even.} \end{cases}$$

- Next, we have

$$a_1^{n_1}, \dots, a_{2k}^{n_{2k}}, E^* \to D$$

is derivable iff machine $\mathscr{M}$ does not halt when started from $(q_0; n_1, \dots, n_{2k}, 0, 0, 0)$.

Multiplexing: Lower Bound

- The quantifier is encoded as follows:

$$K_m = \begin{cases} p_{m-1} \multimap (a_m^+ \cdot p_m) & \text{if } m \text{ is odd;} \\ p_{m-1} \multimap !(a_m \cdot !p_m) & \text{if } m \text{ is even.} \end{cases}$$

- Next, we have

$$a_1^{n_1}, \dots, a_{2k}^{n_{2k}}, E^* \to D$$

  is derivable iff machine $\mathscr{M}$ does not halt when started from $(q_0; n_1, \dots, n_{2k}, 0, 0, 0)$.
- Now we add the quantifier prefix:

$$K_1, \dots, K_{2k}, E^* \to D$$

Multiplexing: Lower Bound

- The quantifier is encoded as follows:

$$K_m = \begin{cases} p_{m-1} \multimap (a_m^+ \cdot p_m) & \text{if } m \text{ is odd;} \\ p_{m-1} \multimap !(a_m \cdot !p_m) & \text{if } m \text{ is even.} \end{cases}$$

- Next, we have

$$a_1^{n_1}, \dots, a_{2k}^{n_{2k}}, E^* \to D$$

  is derivable iff machine $\mathscr{M}$ does not halt when started from $(q_0; n_1, \dots, n_{2k}, 0, 0, 0)$.

- Now we add the quantifier prefix:

$$K_1, \dots, K_{2k}, E^* \to D$$

- The "keys" p_i guarantee that * and ! are decomposed in the correct order.

Multiplexing: Lower Bound

- For the non-commutative case, we use ∇ to encode the commutative system.

Multiplexing: Lower Bound

- For the non-commutative case, we use ∇ to encode the commutative system.
- Complexity without ∇ remains an open question.

Multiplexing: Lower Bound

- For the non-commutative case, we use ∇ to encode the commutative system.
- Complexity without ∇ remains an open question.
- Thus, for systems with multiplexing we have non-trivial upper and lower bound, but they still do not match.

Multiplexing: Lower Bound

- For the non-commutative case, we use ∇ to encode the commutative system.
- Complexity without ∇ remains an open question.
- Thus, for systems with multiplexing we have non-trivial upper and lower bound, but they still do not match.
- It could be the case that the real complexity class is properly hyperarithmetic.

Independence Condition

- Let us return to the system with contraction and consider yet another fragment with intermediate complexity.

Independence Condition

- Let us return to the system with contraction and consider yet another fragment with intermediate complexity.
- This fragment is delimited by the so-called **independence condition:** no formula of the form A^* appears inside a formula of the form $!^i B$, where $i \in \mathscr{C}$.

Independence Condition

- Let us return to the system with contraction and consider yet another fragment with intermediate complexity.
- This fragment is delimited by the so-called **independence condition:** no formula of the form A^* appears inside a formula of the form $!^i B$, where $i \in \mathscr{C}$.

Theorem (K. 2021)

The fragment of $\mathbf{!ACT}_\omega$ *with the independence condition imposed belongs to* Δ^1_1 *and is* Π^0_2*-hard.*

Independence Condition

- Let us return to the system with contraction and consider yet another fragment with intermediate complexity.
- This fragment is delimited by the so-called **independence condition:** no formula of the form A^* appears inside a formula of the form $!^i B$, where $i \in \mathscr{C}$.

Theorem (K. 2021)

The fragment of $\mathbf{!ACT}_\omega$ *with the independence condition imposed belongs to* Δ^1_1 *and is* Π^0_2*-hard.*

- The lower bound could probably be raised to arithmetical.

Independence Condition

- Let us return to the system with contraction and consider yet another fragment with intermediate complexity.
- This fragment is delimited by the so-called **independence condition:** no formula of the form A^* appears inside a formula of the form $!^i B$, where $i \in \mathscr{C}$.

Theorem (K. 2021)

The fragment of $\mathbf{!ACT}_\omega$ *with the independence condition imposed belongs to* Δ^1_1 *and is* Π^0_2*-hard.*

- The lower bound could probably be raised to arithmetical.
- For the upper one, we use *non-well-founded proofs.*

Independence Condition

- Let us return to the system with contraction and consider yet another fragment with intermediate complexity.
- This fragment is delimited by the so-called **independence condition:** no formula of the form A^* appears inside a formula of the form $!^i B$, where $i \in \mathscr{C}$.

Theorem (K. 2021)

The fragment of $\mathbf{!ACT}_\omega$ *with the independence condition imposed belongs to* Δ^1_1 *and is* Π^0_2*-hard.*

- The lower bound could probably be raised to arithmetical.
- For the upper one, we use *non-well-founded proofs.*
- As there are no "Δ^1_1-complete" problems, the upper bound should also be lowered.

Non-well-founded Proofs

- The rules for * are now as follows:

$$\frac{\Gamma \to C \quad \Gamma, A, A^* \to C}{\Gamma, A^* \to C} *L \qquad \frac{}{\to A^*} *R0 \qquad \frac{\Pi \to A \quad \Delta \to A^*}{\Pi, \Delta \to A^*} *Rn$$

Non-well-founded Proofs

- The rules for * are now as follows:

$$\frac{\Gamma \to C \quad \Gamma, A, A^* \to C}{\Gamma, A^* \to C}\ *L \qquad \frac{}{\to A^*}\ *R0 \qquad \frac{\Pi \to A \quad \Delta \to A^*}{\Pi, \Delta \to A^*}\ *Rn$$

- We allow *infinite direction branches.*

Non-well-founded Proofs

- The rules for * are now as follows:

$$\frac{\Gamma \to C \quad \Gamma, A, A^* \to C}{\Gamma, A^* \to C} *L \qquad \frac{}{\to A^*} *R0 \qquad \frac{\Pi \to A \quad \Delta \to A^*}{\Pi, \Delta \to A^*} *Rn$$

- We allow *infinite direction branches.*
- **Correctness criterion** *(*-fairness):* each infinite branch should infinitely often traverse $*L$ to the right.

Non-well-founded Proofs

- The rules for * are now as follows:

$$\frac{\Gamma \to C \quad \Gamma, A, A^* \to C}{\Gamma, A^* \to C} \; *L \qquad \frac{}{\to A^*} \; *R0 \qquad \frac{\Pi \to A \quad \Delta \to A^*}{\Pi, \Delta \to A^*} \; *Rn$$

- We allow *infinite direction branches.*
- **Correctness criterion** *(*-fairness):* each infinite branch should infinitely often traverse $*L$ to the right.
- This yields a Σ^1_1 upper bound, thus we get Δ^1_1 upper bound (hyperarithmetical).

Non-well-founded Proofs

- The rules for * are now as follows:

$$\frac{\Gamma \to C \quad \Gamma, A, A^* \to C}{\Gamma, A^* \to C} \ast L \qquad \frac{}{\to A^*} \ast R0 \qquad \frac{\Pi \to A \quad \Delta \to A^*}{\Pi, \Delta \to A^*} \ast Rn$$

- We allow *infinite direction branches.*
- **Correctness criterion** *(*-fairness):* each infinite branch should infinitely often traverse $\ast L$ to the right.
- This yields a Σ^1_1 upper bound, thus we get Δ^1_1 upper bound (hyperarithmetical).
- A lower bound of Π^0_2 could also be proved.

Non-well-founded Proofs

- If we manage to make the infinite proof *computable*, we would get even Σ^0_3.

Non-well-founded Proofs

- If we manage to make the infinite proof *computable*, we would get even Σ^0_3.
- Independence condition is crucial: otherwise our correctness criterion is too weak.

Non-well-founded Proofs

- If we manage to make the infinite proof *computable*, we would get even Σ^0_3.
- Independence condition is crucial: otherwise our correctness criterion is too weak.
  - Counterexample (in the non-commutative case):

$$s, !(s \multimap (p^* \cdot ((p^+ \multimap \mathbf{1}) \wedge q) \cdot s)) \rightarrow q \cdot s$$

Non-well-founded Proofs

- If we manage to make the infinite proof *computable*, we would get even Σ^0_3.
- Independence condition is crucial: otherwise our correctness criterion is too weak.
  - Counterexample (in the non-commutative case):

$$s, !(s \multimap (p^* \cdot ((p^+ \multimap \mathbf{1}) \wedge q) \cdot s)) \to q \cdot s$$

- We conjecture that the stronger criterion would work: on each infinite branch, there is a trace of *the same* A^*.

Non-well-founded Proofs

- If we manage to make the infinite proof *computable,* we would get even Σ^0_3.
- Independence condition is crucial: otherwise our correctness criterion is too weak.
  - Counterexample (in the non-commutative case):

$$s, !(s \multimap (p^* \cdot ((p^+ \multimap \mathbf{1}) \wedge q) \cdot s)) \to q \cdot s$$

- We conjecture that the stronger criterion would work: on each infinite branch, there is a trace of *the same* A^*.
- However, this does not give complexity gain.

Thanks*