# Infinitary Substructural Logics

Stepan Kuznetsov

Steklov Mathematical Institute

20th International Tbilisi Summer School in Logic and Language
Tbilisi, September 2026

# Substructural Logics

- This course considers certain non-classical logics used for modelling language and computation.

# Substructural Logics

- This course considers certain non-classical logics used for modelling language and computation.
- **Substructural logics** are logical systems which lack some or all of the **structural rules.**

# Substructural Logics

- This course considers certain non-classical logics used for modelling language and computation.
- **Substructural logics** are logical systems which lack some or all of the **structural rules.**
- **Contraction:** $A \to A \,\&\, A$, or $(A \to (A \to B)) \to (A \to B)$.

# Substructural Logics

- This course considers certain non-classical logics used for modelling language and computation.
- **Substructural logics** are logical systems which lack some or all of the **structural rules.**
- **Contraction:** $A \to A \,\&\, A$, or $(A \to (A \to B)) \to (A \to B)$.
  - Allows copying a formula.

# Substructural Logics

- This course considers certain non-classical logics used for modelling language and computation.
- **Substructural logics** are logical systems which lack some or all of the **structural rules.**
- **Contraction:** $A \to A \,\&\, A$, or $(A \to (A \to B)) \to (A \to B)$.
  - Allows copying a formula.
- **Weakening:** $A \to \top$, or $B \to (A \to B)$.

# Substructural Logics

- This course considers certain non-classical logics used for modelling language and computation.
- **Substructural logics** are logical systems which lack some or all of the **structural rules.**
- **Contraction:** $A \to A \,\&\, A$, or $(A \to (A \to B)) \to (A \to B)$.
  - Allows copying a formula.
- **Weakening:** $A \to \top$, or $B \to (A \to B)$.
  - Allows dismissing a formula.

# Substructural Logics

- This course considers certain non-classical logics used for modelling language and computation.
- **Substructural logics** are logical systems which lack some or all of the **structural rules.**
- **Contraction:** $A \to A \,\&\, A$, or $(A \to (A \to B)) \to (A \to B)$.
  - Allows copying a formula.
- **Weakening:** $A \to \top$, or $B \to (A \to B)$.
  - Allows dismissing a formula.
- **Exchange:** $A \,\&\, B \to B \,\&\, A$, or $(B \to (A \to C)) \to (A \to (B \to C))$.

# Substructural Logics

- This course considers certain non-classical logics used for modelling language and computation.
- **Substructural logics** are logical systems which lack some or all of the **structural rules.**
- **Contraction:** $A \to A \,\&\, A$, or $(A \to (A \to B)) \to (A \to B)$.
  - Allows copying a formula.
- **Weakening:** $A \to \top$, or $B \to (A \to B)$.
  - Allows dismissing a formula.
- **Exchange:** $A \,\&\, B \to B \,\&\, A$, or $(B \to (A \to C)) \to (A \to (B \to C))$.
  - The order of formulae does not matter.

# Substructural Logics

- This course considers certain non-classical logics used for modelling language and computation.
- **Substructural logics** are logical systems which lack some or all of the **structural rules.**
- **Contraction:** $A \to A \,\&\, A$, or $(A \to (A \to B)) \to (A \to B)$.
  - Allows copying a formula.
- **Weakening:** $A \to \top$, or $B \to (A \to B)$.
  - Allows dismissing a formula.
- **Exchange:** $A \,\&\, B \to B \,\&\, A$, or $(B \to (A \to C)) \to (A \to (B \to C))$.
  - The order of formulae does not matter.
- These principles are readily available in classical and intuitionistic logics.

# Substructural Logics

- This course considers certain non-classical logics used for modelling language and computation.
- **Substructural logics** are logical systems which lack some or all of the **structural rules.**
- **Contraction:** $A \to A \,\&\, A$, or $(A \to (A \to B)) \to (A \to B)$.
  - Allows copying a formula.
- **Weakening:** $A \to \top$, or $B \to (A \to B)$.
  - Allows dismissing a formula.
- **Exchange:** $A \,\&\, B \to B \,\&\, A$, or $(B \to (A \to C)) \to (A \to (B \to C))$.
  - The order of formulae does not matter.
- These principles are readily available in classical and intuitionistic logics.
- Reading formulae as **resources** [Girard 1987], however, refutes contraction and weakening.

# Substructural Logics

- This course considers certain non-classical logics used for modelling language and computation.
- **Substructural logics** are logical systems which lack some or all of the **structural rules.**
- **Contraction:** $A \to A \,\&\, A$, or $(A \to (A \to B)) \to (A \to B)$.
  - Allows copying a formula.
- **Weakening:** $A \to \top$, or $B \to (A \to B)$.
  - Allows dismissing a formula.
- **Exchange:** $A \,\&\, B \to B \,\&\, A$, or $(B \to (A \to C)) \to (A \to (B \to C))$.
  - The order of formulae does not matter.
- These principles are readily available in classical and intuitionistic logics.
- Reading formulae as **resources** [Girard 1987], however, refutes contraction and weakening.
- Non-commutative (without exchange) resources are **actions.**

# Substructural Logics

- All three structural rules are removed in the **Lambek calculus** [Lambek 1958], whose formulae are used for modelling syntactic objects in linguistics and actions in computer science.

# Substructural Logics

- All three structural rules are removed in the **Lambek calculus** [Lambek 1958], whose formulae are used for modelling syntactic objects in linguistics and actions in computer science.
- Removing weakening, while keeping contraction (and, possibly, exchange), leads to the **relevant logic** tradition.

# Substructural Logics

- All three structural rules are removed in the **Lambek calculus** [Lambek 1958], whose formulae are used for modelling syntactic objects in linguistics and actions in computer science.
- Removing weakening, while keeping contraction (and, possibly, exchange), leads to the **relevant logic** tradition.
- We shall use **sequent calculi.** A sequent is an object of the form $A_1, \ldots, A_n \vdash B$, meaning that the formula $B$ is entailed by $A_1, \ldots, A_n$ (in the given order and quantity).

# Substructural Logics

- All three structural rules are removed in the **Lambek calculus** [Lambek 1958], whose formulae are used for modelling syntactic objects in linguistics and actions in computer science.
- Removing weakening, while keeping contraction (and, possibly, exchange), leads to the **relevant logic** tradition.
- We shall use **sequent calculi.** A sequent is an object of the form $A_1, \ldots, A_n \vdash B$, meaning that the formula $B$ is entailed by $A_1, \ldots, A_n$ (in the given order and quantity).
- In the sequent formalism, structural rules are represented as follows:

$$\frac{\Gamma, A, A, \Delta \vdash C}{\Gamma, A, \Delta \vdash C}\ \mathsf{C} \qquad \frac{\Gamma, \Delta \vdash C}{\Gamma, A, \Delta \vdash C}\ \mathsf{W} \qquad \frac{\Gamma, A, B, \Delta \vdash C}{\Gamma, B, A, \Delta \vdash C}\ \mathsf{E}$$

(Where $\Gamma$ and $\Delta$ are sequences of formulae.)

# Logical Rules

- The usual sequent calculus rules (in intuitionistic logic) for implication are as follows [Gentzen 1934]:

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

# Logical Rules

- The usual sequent calculus rules (in intuitionistic logic) for implication are as follows [Gentzen 1934]:

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

  - The first one corresponds to *modus ponens*: $A, A \to B \vdash B$; the second one is deduction theorem.

# Logical Rules

- The usual sequent calculus rules (in intuitionistic logic) for implication are as follows [Gentzen 1934]:

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

  - The first one corresponds to *modus ponens*: $A, A \to B \vdash B$; the second one is deduction theorem.

- In the non-commutative setting (no exchange rule), implication is **directed,** left or right:

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

$$\frac{\Pi \vdash A \quad \Gamma, B, \Delta \vdash C}{\Gamma, B \leftarrow A, \Pi, \Delta \vdash C} \leftarrow\mathsf{L} \qquad \frac{\Pi, A \vdash B}{\Pi \vdash B \leftarrow A} \leftarrow\mathsf{R}$$

# Logical Rules

- The usual sequent calculus rules (in intuitionistic logic) for implication are as follows [Gentzen 1934]:

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

  - The first one corresponds to *modus ponens*: $A, A \to B \vdash B$; the second one is deduction theorem.

- In the non-commutative setting (no exchange rule), implication is **directed,** left or right:

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

$$\frac{\Pi \vdash A \quad \Gamma, B, \Delta \vdash C}{\Gamma, B \leftarrow A, \Pi, \Delta \vdash C} \leftarrow\mathsf{L} \qquad \frac{\Pi, A \vdash B}{\Pi \vdash B \leftarrow A} \leftarrow\mathsf{R}$$

  - In Lambek's tradition, directed implications are denoted as **divisions:** $\backslash, /$.

# Logical Rules

- The usual sequent calculus rules (in intuitionistic logic) for implication are as follows [Gentzen 1934]:

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

  - The first one corresponds to *modus ponens*: $A, A \to B \vdash B$; the second one is deduction theorem.

- In the non-commutative setting (no exchange rule), implication is **directed,** left or right:

$$\frac{\Pi \vdash A \quad \Gamma, B, \Delta \vdash C}{\Gamma, \Pi, A \setminus B, \Delta \vdash C} \setminus\mathsf{L} \qquad \frac{A, \Pi \vdash B}{\Pi \vdash A \setminus B} \setminus\mathsf{R}$$

$$\frac{\Pi \vdash A \quad \Gamma, B, \Delta \vdash C}{\Gamma, B / A, \Pi, \Delta \vdash C} /\mathsf{L} \qquad \frac{\Pi, A \vdash B}{\Pi \vdash B / A} /\mathsf{R}$$

  - In Lambek's tradition, directed implications are denoted as **divisions:** $\setminus, /$.

# Logical Rules

- Similarly, in the absence of contraction we have two versions of conjunction: $\otimes$ (**multiplicative conjunction,** or **product**) and $\wedge$ (**additive conjunction,** or **meet**):

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

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

# Logical Rules

- Similarly, in the absence of contraction we have two versions of conjunction: $\otimes$ (**multiplicative conjunction,** or **product**) and $\wedge$ (**additive conjunction,** or **meet**):

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

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

- In the resouce understanding, $A \otimes B$ means a bundle of two resources (in the given order), while $A \wedge B$ is one resource which can act both as $A$ and as $B$.

# Logical Rules

- Similarly, in the absence of contraction we have two versions of conjunction: $\otimes$ (**multiplicative conjunction,** or **product**) and $\wedge$ (**additive conjunction,** or **meet**):

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

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

- In the resouce understanding, $A \otimes B$ means a bundle of two resources (in the given order), while $A \wedge B$ is one resource which can act both as $A$ and as $B$.
- Contraction makes $A \otimes B$ and $A \wedge B$ equivalent.

# The Lambek Calculus

- Our minimal associative substructural logic, in the multiplicative language $(\backslash, /, \otimes)$ is the **Lambek calculus** **L**:

$$\frac{}{A \vdash A}\ \mathsf{Id} \qquad \frac{\Pi \vdash A \quad \Gamma, A, \Delta \vdash C}{\Gamma, \Pi, \Delta \vdash C}\ \mathsf{Cut}$$

$$\frac{\Pi \vdash A \quad \Gamma, B, \Delta \vdash C}{\Gamma, \Pi, A \backslash B, \Delta \vdash C}\ \backslash\mathsf{L} \qquad \frac{A, \Pi \vdash B}{\Pi \vdash A \backslash B}\ \backslash\mathsf{R} \qquad \frac{\Gamma, A, B, \Delta \vdash C}{\Gamma, A \otimes B, \Delta \vdash C}\ \otimes\mathsf{L}$$

$$\frac{\Pi \vdash A \quad \Gamma, B, \Delta \vdash C}{\Gamma, B / A, \Pi, \Delta \vdash C}\ /\mathsf{L} \qquad \frac{\Pi, A \vdash B}{\Pi \vdash B / A}\ /\mathsf{R} \qquad \frac{\Pi \vdash A \quad \Delta \vdash B}{\Pi, \Delta \vdash A \otimes B}\ \otimes\mathsf{R}$$

# The Lambek Calculus

- Our minimal associative substructural logic, in the multiplicative language $(\backslash, /, \otimes)$ is the **Lambek calculus** **L**:

$$\frac{}{A \vdash A}\ \mathsf{Id} \qquad \frac{\Pi \vdash A \quad \Gamma, A, \Delta \vdash C}{\Gamma, \Pi, \Delta \vdash C}\ \mathsf{Cut}$$

$$\frac{\Pi \vdash A \quad \Gamma, B, \Delta \vdash C}{\Gamma, \Pi, A \backslash B, \Delta \vdash C}\ \backslash\mathsf{L} \qquad \frac{A, \Pi \vdash B}{\Pi \vdash A \backslash B}\ \backslash\mathsf{R} \qquad \frac{\Gamma, A, B, \Delta \vdash C}{\Gamma, A \otimes B, \Delta \vdash C}\ \otimes\mathsf{L}$$

$$\frac{\Pi \vdash A \quad \Gamma, B, \Delta \vdash C}{\Gamma, B / A, \Pi, \Delta \vdash C}\ /\mathsf{L} \qquad \frac{\Pi, A \vdash B}{\Pi \vdash B / A}\ /\mathsf{R} \qquad \frac{\Pi \vdash A \quad \Delta \vdash B}{\Pi, \Delta \vdash A \otimes B}\ \otimes\mathsf{R}$$

- The Cut rule is eliminable.

# The Lambek Calculus

- Our minimal associative substructural logic, in the multiplicative language $(\backslash, /, \otimes)$ is the **Lambek calculus** **L**:

$$\frac{}{A \vdash A}\ \mathsf{Id} \qquad \frac{\Pi \vdash A \quad \Gamma, A, \Delta \vdash C}{\Gamma, \Pi, \Delta \vdash C}\ \mathsf{Cut}$$

$$\frac{\Pi \vdash A \quad \Gamma, B, \Delta \vdash C}{\Gamma, \Pi, A \backslash B, \Delta \vdash C}\ \backslash\mathsf{L} \qquad \frac{A, \Pi \vdash B}{\Pi \vdash A \backslash B}\ \backslash\mathsf{R} \qquad \frac{\Gamma, A, B, \Delta \vdash C}{\Gamma, A \otimes B, \Delta \vdash C}\ \otimes\mathsf{L}$$

$$\frac{\Pi \vdash A \quad \Gamma, B, \Delta \vdash C}{\Gamma, B / A, \Pi, \Delta \vdash C}\ /\mathsf{L} \qquad \frac{\Pi, A \vdash B}{\Pi \vdash B / A}\ /\mathsf{R} \qquad \frac{\Pi \vdash A \quad \Delta \vdash B}{\Pi, \Delta \vdash A \otimes B}\ \otimes\mathsf{R}$$

- The Cut rule is eliminable.
- In the original formulation of **L**, left-hand sides were required to be non-empty (Lambek's non-emptiness restriction); we do not impose this restiction.

# The Full Lambek Calculus

- The **multiplicative-additive,** or **full** Lambek calculus **FL** is obtained from **L** by adding additive conjunction and disjunction and constants:

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

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

$$\frac{}{\Gamma, 0, \Delta \vdash C}\ 0\mathsf{L} \qquad \frac{\Gamma, \Delta \vdash C}{\Gamma, 1, \Delta \vdash C}\ 1\mathsf{L} \qquad \frac{}{\vdash 1}\ 1\mathsf{R}$$

# The Full Lambek Calculus

- The **multiplicative-additive,** or **full** Lambek calculus **FL** is obtained from **L** by adding additive conjunction and disjunction and constants:

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

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

$$\frac{}{\Gamma, 0, \Delta \vdash C}\ 0\mathsf{L} \qquad \frac{\Gamma, \Delta \vdash C}{\Gamma, 1, \Delta \vdash C}\ 1\mathsf{L} \qquad \frac{}{\vdash 1}\ 1\mathsf{R}$$

- **FL** can be further extended by structural rules: e.g., $\mathbf{FL}_{\mathrm{ec}}$ is the non-distributive relevant logic.

# Lambek Grammars

- The linguistic interpretation of the Lambek calculus considers Lambek formulae as **syntactic types** (categories).

# Lambek Grammars

- The linguistic interpretation of the Lambek calculus considers Lambek formulae as **syntactic types** (categories).
- $A \setminus B$ (resp., $B / A$) means "something which lacks $A$ on the left (resp., on the right) to become $B$."

# Lambek Grammars

- The linguistic interpretation of the Lambek calculus considers Lambek formulae as **syntactic types** (categories).
- $A \setminus B$ (resp., $B / A$) means "something which lacks $A$ on the left (resp., on the right) to become $B$."
- Syntactic types are associated to words (lexemes) in the **lexicon** of a Lambek categorial grammar.

# Lambek Grammars

- The linguistic interpretation of the Lambek calculus considers Lambek formulae as **syntactic types** (categories).
- $A \setminus B$ (resp., $B / A$) means "something which lacks $A$ on the left (resp., on the right) to become $B$."
- Syntactic types are associated to words (lexemes) in the **lexicon** of a Lambek categorial grammar.
- Having $np$ (noun phrase), $n$ (common noun), and $s$ (sentence) as basic (primitive) types, we can start building our lexicon:

$$\begin{array}{rr} \textit{John}, \textit{Mary} \rhd np & \textit{loves} \rhd (np \setminus s) / np \\ \textit{girl} \rhd n & \textit{smiles} \rhd np \setminus s \\ \textit{the} \rhd np / n & \textit{charmingly} \rhd (np \setminus s) \setminus (np \setminus s) \end{array}$$

# Lambek Grammars

- Grammatical correctness of sentences is validated by derivability of the corresponding sequents in the Lambek calculus:

| | |
|---|---|
| *Mary smiles charmingly* | $np, np \setminus s, (np \setminus s) \setminus (np \setminus s) \vdash s$ |
| *John loves Mary* | $np, (np \setminus s) / np, np \vdash s$ |

# Lambek Grammars

- Grammatical correctness of sentences is validated by derivability of the corresponding sequents in the Lambek calculus:

| | |
|---|---|
| *Mary smiles charmingly* | $np, np \setminus s, (np \setminus s) \setminus (np \setminus s) \vdash s$ |
| *John loves Mary* | $np, (np \setminus s) / np, np \vdash s$ |

- The right rules ($\setminus$R, /R) are used for *hypothetical reasoning* which models dependent clauses:

  *the girl whom John loves*
  $np / n, n, (n \setminus n) / (s / np), np, (np \setminus s) / np \vdash np,$

  where *whom* $\triangleright (n \setminus n) / (s / np)$.

# Lambek Grammars

- Grammatical correctness of sentences is validated by derivability of the corresponding sequents in the Lambek calculus:

| | |
|---|---|
| *Mary smiles charmingly* | $np, np \setminus s, (np \setminus s) \setminus (np \setminus s) \vdash s$ |
| *John loves Mary* | $np, (np \setminus s) / np, np \vdash s$ |

- The right rules ($\setminus$R, /R) are used for *hypothetical reasoning* which models dependent clauses:

  *the girl whom John loves*
  $np / n, n, (n \setminus n) / (s / np), np, (np \setminus s) / np \vdash np,$

  where *whom* $\triangleright (n \setminus n) / (s / np)$.
- Word order matters (at least in English), so we do not have exchange.

# Lambek Grammars

- Lambek grammars have certain limitations, which motivate considering extended systems.

# Lambek Grammars

- Lambek grammars have certain limitations, which motivate considering extended systems.
- For example, $whom \triangleright (n \setminus n) / (s / np)$ is insufficient to model

*the girl whom John met yesterday.*

# Lambek Grammars

- Lambek grammars have certain limitations, which motivate considering extended systems.
- For example, $whom \triangleright (n \setminus n) / (s / np)$ is insufficient to model

  *the girl whom John met yesterday.*

- A theoretical argument: Lambek grammars generate only context-free languages [Pentus 1992], while natural language (arguably) exhibits non-context-free behaviour [Shieber 1985].

# Lambek Grammars

- Lambek grammars have certain limitations, which motivate considering extended systems.
- For example, $whom \rhd (n \setminus n) / (s / np)$ is insufficient to model

  *the girl whom John met yesterday.*

- A theoretical argument: Lambek grammars generate only context-free languages [Pentus 1992], while natural language (arguably) exhibits non-context-free behaviour [Shieber 1985].
- Example from Old Georgian [Michaelis, Kracht 1996]:

| *govel-i* | *igi* | *sisxl-i* | *saxl-isa-j* | *m-is* | *Saul-is-isa-j* |
|---|---|---|---|---|---|
| all-Nom | Art=Nom | blood-Nom | house-Gen-Nom | Art-Gen | Saul-Gen-Gen-Nom |

*'all the blood of the house of Saul'*

For $k$ stacked $np$'s, the number of genitive suffixes (*isa*, *is*) is bounded by $\frac{k(k-1)}{2}$, which is not semilinear.

# Kleene Star

- We shall focus on **iteration,** or **Kleene star** as an additional operation in the Lambek calculus.

# Kleene Star

- We shall focus on **iteration,** or **Kleene star** as an additional operation in the Lambek calculus.
- Example: iterated coordination [Morrill 1994, 2011] in

  $$\textit{John, Bill, Mary, and Suzy,}$$

  where $\textit{and} \rhd (np^{+} \setminus np) \,/\, np$.

# Kleene Star

- We shall focus on **iteration,** or **Kleene star** as an additional operation in the Lambek calculus.
- Example: iterated coordination [Morrill 1994, 2011] in

  $$\textit{John, Bill, Mary, and Suzy,}$$

  where $\textit{and} \rhd (np^+ \setminus np) \,/\, np$.
- Here $np^+ = np \otimes np^*$ denotes "any positive number of $np$'s."

# Kleene Star

- We shall focus on **iteration,** or **Kleene star** as an additional operation in the Lambek calculus.
- Example: iterated coordination [Morrill 1994, 2011] in

  $$\textit{John, Bill, Mary, and Suzy,}$$

  where $\textit{and} \rhd (np^+ \setminus np) \,/\, np$.
- Here $np^+ = np \otimes np^*$ denotes "any positive number of $np$'s."
- Another example: an alternative account of adjectives [Buszkowski, Palka 2008]: $\quad \textit{big}, \textit{red} \rhd a; \quad \textit{ball} \rhd a^* \setminus n.$

# Kleene Star

- We shall focus on **iteration,** or **Kleene star** as an additional operation in the Lambek calculus.
- Example: iterated coordination [Morrill 1994, 2011] in

  $$\textit{John, Bill, Mary, and Suzy,}$$

  where $\textit{and} \rhd (np^+ \setminus np) \,/\, np$.
- Here $np^+ = np \otimes np^*$ denotes "any positive number of $np$'s."
- Another example: an alternative account of adjectives [Buszkowski, Palka 2008]: $\textit{big}, \textit{red} \rhd a$; $\textit{ball} \rhd a^* \setminus n$.
  - More traditionally: $\textit{ball} \rhd n$; $\textit{big}, \textit{red} \rhd n \,/\, n$.

# Kleene Star

- We shall focus on **iteration,** or **Kleene star** as an additional operation in the Lambek calculus.
- Example: iterated coordination [Morrill 1994, 2011] in

  $$\textit{John, Bill, Mary, and Suzy,}$$

  where $\textit{and} \rhd (np^+ \setminus np) \,/\, np$.
- Here $np^+ = np \otimes np^*$ denotes "any positive number of $np$'s."
- Another example: an alternative account of adjectives [Buszkowski, Palka 2008]: $\textit{big}, \textit{red} \rhd a$; $\textit{ball} \rhd a^* \setminus n$.
  - More traditionally: $\textit{ball} \rhd n$; $\textit{big}, \textit{red} \rhd n \,/\, n$.
  - The alternative way allows more fine-grained control over adjectives. For example, with

    $$\textit{tall} \rhd a_R; \quad \textit{American} \rhd a_A; \quad \textit{student} \rhd (a_R^* \otimes a_A^*) \setminus n,$$

    one validates *tall American student* and invalidates *American tall student*.

# Kleene Star

- In the aforementioned examples, the following right rule for Kleene star is used:

$$\frac{\Pi_1 \vdash A \quad \ldots \quad \Pi_n \vdash A}{\Pi_1, \ldots, \Pi_n \vdash A^*} \; *\mathrm{R}_n$$

# Kleene Star

- In the aforementioned examples, the following right rule for Kleene star is used:

$$\frac{\Pi_1 \vdash A \quad \ldots \quad \Pi_n \vdash A}{\Pi_1, \ldots, \Pi_n \vdash A^*} \; *\mathrm{R}_n$$

  - In fact, it is an infinitary series of rules, for each $n \geq 0$.

# Kleene Star

- In the aforementioned examples, the following right rule for Kleene star is used:

$$\frac{\Pi_1 \vdash A \quad \ldots \quad \Pi_n \vdash A}{\Pi_1, \ldots, \Pi_n \vdash A^*} *\mathsf{R}_n$$

  - In fact, it is an infinitary series of rules, for each $n \geq 0$.
- Example: derivation for *John, Bill, Mary, and Suzy:*

$$\dfrac{np \vdash np \qquad \dfrac{\dfrac{np \vdash np \quad np \vdash np}{np, np \vdash np^*}\, *\mathsf{R}_2 \qquad np \vdash np}{np, np, np \vdash np \otimes np^*}\, \otimes\mathsf{R} \qquad np \vdash np}{np, np, np, (np \otimes np^*) \setminus np \vdash np}\, \setminus\mathsf{L}}{np, np, np, ((np \otimes np^*) \setminus np) \,/\, np, np \vdash np}\, /\mathsf{L}$$

# Kleene Star

- What about the left rule?

# Kleene Star

- What about the left rule?
- In order to validate $A^* \vdash B$, one needs $A^n \vdash B$ **for all** $n \geq 0$.

# Kleene Star

- What about the left rule?
- In order to validate $A^* \vdash B$, one needs $A^n \vdash B$ **for all** $n \geq 0$.
- This suggests the following **$\omega$-rule** (infinitary rule):

$$\frac{\Gamma, \Delta \vdash B \quad \Gamma, A, \Delta \vdash B \quad \Gamma, A, A, \Delta \vdash B \quad \Gamma, A, A, A, \Delta \vdash B \quad \ldots}{\Gamma, A^*, \Delta \vdash B} \; *\mathsf{L}_\omega$$

In a more compact form:

$$\frac{\left(\Gamma, A^n, \Delta \vdash B\right)_{n=0}^{\infty}}{\Gamma, A^*, \Delta \vdash B} \; *\mathsf{L}_\omega$$

# Kleene Star

- What about the left rule?
- In order to validate $A^* \vdash B$, one needs $A^n \vdash B$ **for all** $n \geq 0$.
- This suggests the following **$\omega$-rule** (infinitary rule):

$$\frac{\Gamma, \Delta \vdash B \quad \Gamma, A, \Delta \vdash B \quad \Gamma, A, A, \Delta \vdash B \quad \Gamma, A, A, A, \Delta \vdash B \quad \ldots}{\Gamma, A^*, \Delta \vdash B} *\mathsf{L}_\omega$$

In a more compact form:

$$\frac{\left(\Gamma, A^n, \Delta \vdash B\right)_{n=0}^{\infty}}{\Gamma, A^*, \Delta \vdash B} *\mathsf{L}_\omega$$

- Derivations are now **infinitely branching** trees.

# Kleene Star

- What about the left rule?
- In order to validate $A^* \vdash B$, one needs $A^n \vdash B$ **for all** $n \geq 0$.
- This suggests the following **$\omega$-rule** (infinitary rule):

$$\frac{\Gamma, \Delta \vdash B \quad \Gamma, A, \Delta \vdash B \quad \Gamma, A, A, \Delta \vdash B \quad \Gamma, A, A, A, \Delta \vdash B \quad \ldots}{\Gamma, A^*, \Delta \vdash B} *\mathsf{L}_\omega$$

In a more compact form:

$$\frac{\left(\Gamma, A^n, \Delta \vdash B\right)_{n=0}^{\infty}}{\Gamma, A^*, \Delta \vdash B} *\mathsf{L}_\omega$$

- Derivations are now **infinitely branching** trees.
- Infinite *upward paths*, however, are still disallowed (**well-foundedness**).

# Infinitary Action Logic

- **FL** extended with $*\mathsf{R}_n$ and $*\mathsf{L}_\omega$ is **infinitary action logic** $\mathbf{ACT}^\omega$ [Palka 2007].

$$\frac{\left(\Gamma, A^n, \Delta \vdash B\right)_{n=0}^{\infty}}{\Gamma, A^*, \Delta \vdash B}\ *\mathsf{L}_\omega \qquad \frac{\Pi_1 \vdash A \quad \ldots \quad \Pi_n \vdash A}{\Pi_1, \ldots, \Pi_n \vdash A^*}\ *\mathsf{R}_n,\ n \geq 0$$

# Infinitary Action Logic

- **FL** extended with $*\mathsf{R}_n$ and $*\mathsf{L}_\omega$ is **infinitary action logic** $\mathbf{ACT}^\omega$ [Palka 2007].

$$\frac{\left(\Gamma, A^n, \Delta \vdash B\right)_{n=0}^{\infty}}{\Gamma, A^*, \Delta \vdash B}\ *\mathsf{L}_\omega \qquad \frac{\Pi_1 \vdash A \quad \ldots \quad \Pi_n \vdash A}{\Pi_1, \ldots, \Pi_n \vdash A^*}\ *\mathsf{R}_n,\ n \geq 0$$

- In linguistic applications, $*\mathsf{L}_\omega$ is never used. (Kleene star appears only in positive positions.)

# Infinitary Action Logic

- **FL** extended with $*\mathsf{R}_n$ and $*\mathsf{L}_\omega$ is **infinitary action logic** $\mathbf{ACT}^\omega$ [Palka 2007].

$$\frac{\left(\Gamma, A^n, \Delta \vdash B\right)_{n=0}^{\infty}}{\Gamma, A^*, \Delta \vdash B}\ *\mathsf{L}_\omega \qquad \frac{\Pi_1 \vdash A \quad \ldots \quad \Pi_n \vdash A}{\Pi_1, \ldots, \Pi_n \vdash A^*}\ *\mathsf{R}_n,\ n \geq 0$$

- In linguistic applications, $*\mathsf{L}_\omega$ is never used. (Kleene star appears only in positive positions.)
  - However, it is officially included in the calculus for CatLog3 system [Morrill 2019].

# Infinitary Action Logic

- **FL** extended with $*\mathsf{R}_n$ and $*\mathsf{L}_\omega$ is **infinitary action logic** $\mathbf{ACT}^\omega$ [Palka 2007].

$$\frac{\left(\Gamma, A^n, \Delta \vdash B\right)_{n=0}^{\infty}}{\Gamma, A^*, \Delta \vdash B}\ *\mathsf{L}_\omega \qquad \frac{\Pi_1 \vdash A \quad \ldots \quad \Pi_n \vdash A}{\Pi_1, \ldots, \Pi_n \vdash A^*}\ *\mathsf{R}_n,\ n \geq 0$$

- In linguistic applications, $*\mathsf{L}_\omega$ is never used. (Kleene star appears only in positive positions.)
  - However, it is officially included in the calculus for CatLog3 system [Morrill 2019].
- Another application of $\mathbf{ACT}^\omega$ is modelling **actions** in computation systems.

# Infinitary Action Logic

- **FL** extended with $*\mathsf{R}_n$ and $*\mathsf{L}_\omega$ is **infinitary action logic** $\mathbf{ACT}^\omega$ [Palka 2007].

$$\frac{\left(\Gamma, A^n, \Delta \vdash B\right)_{n=0}^{\infty}}{\Gamma, A^*, \Delta \vdash B}\ *\mathsf{L}_\omega \qquad \frac{\Pi_1 \vdash A \quad \ldots \quad \Pi_n \vdash A}{\Pi_1, \ldots, \Pi_n \vdash A^*}\ *\mathsf{R}_n,\ n \geq 0$$

- In linguistic applications, $*\mathsf{L}_\omega$ is never used. (Kleene star appears only in positive positions.)
  - However, it is officially included in the calculus for CatLog3 system [Morrill 2019].
- Another application of $\mathbf{ACT}^\omega$ is modelling **actions** in computation systems.
  - $\mathbf{ACT}^\omega$ is "**PDL** without propositions" (but with internalised implications).

# Infinitary Action Logic

- This ‘action’ semantics corresponds to interpreting formulae as binary relations over a set of states $W$.

# Infinitary Action Logic

- This ‘action’ semantics corresponds to interpreting formulae as binary relations over a set of states $W$.
- Multiplicative conjunction $\otimes$ is relational composition; Kleene star is the reflexive-transitive closure.

# Infinitary Action Logic

- This 'action' semantics corresponds to interpreting formulae as binary relations over a set of states $W$.
- Multiplicative conjunction $\otimes$ is relational composition; Kleene star is the reflexive-transitive closure.
- Meet $\wedge$ and join $\vee$ are set-theoretic intersection and union.

# Infinitary Action Logic

- This 'action' semantics corresponds to interpreting formulae as binary relations over a set of states $W$.
- Multiplicative conjunction $\otimes$ is relational composition; Kleene star is the reflexive-transitive closure.
- Meet $\wedge$ and join $\vee$ are set-theoretic intersection and union.
- Division on relations is defined as 'conditional action':

$$R \setminus S = \{(y, z) \mid R \circ \{(y, z)\} \subseteq S\},$$
$$S / R = \{(x, y) \mid \{(x, y)\} \circ R \subseteq S\}.$$

# Infinitary Action Logic

- This 'action' semantics corresponds to interpreting formulae as binary relations over a set of states $W$.
- Multiplicative conjunction $\otimes$ is relational composition; Kleene star is the reflexive-transitive closure.
- Meet $\wedge$ and join $\vee$ are set-theoretic intersection and union.
- Division on relations is defined as 'conditional action':

$$R \setminus S = \{(y, z) \mid R \circ \{(y, z)\} \subseteq S\},$$
$$S / R = \{(x, y) \mid \{(x, y)\} \circ R \subseteq S\}.$$

- When reasoning about programs, left rules become necessary.

# Infinitary Action Logic

- This 'action' semantics corresponds to interpreting formulae as binary relations over a set of states $W$.
- Multiplicative conjunction $\otimes$ is relational composition; Kleene star is the reflexive-transitive closure.
- Meet $\wedge$ and join $\vee$ are set-theoretic intersection and union.
- Division on relations is defined as 'conditional action':

$$R \setminus S = \{(y, z) \mid R \circ \{(y, z)\} \subseteq S\},$$
$$S / R = \{(x, y) \mid \{(x, y)\} \circ R \subseteq S\}.$$

- When reasoning about programs, left rules become necessary.
- Example: 'pure induction' [Pratt 1991]

$$(a \setminus a)^* \vdash a \setminus a$$

# Algorithmic Problems

- Theoremhood in the Lambek calculus (both **L** and **FL**) is algorithmically decidable, via exhaustive cut-free proof search.

# Algorithmic Problems

- Theoremhood in the Lambek calculus (both **L** and **FL**) is algorithmically decidable, via exhaustive cut-free proof search.
  - More precisely, it is NP-complete for **L** [Pentus 2006] and PSPACE-complete for **FL** [Kanovich 1994; Kanazawa 1996].

# Algorithmic Problems

- Theoremhood in the Lambek calculus (both **L** and **FL**) is algorithmically decidable, via exhaustive cut-free proof search.
  - More precisely, it is NP-complete for **L** [Pentus 2006] and PSPACE-complete for **FL** [Kanovich 1994; Kanazawa 1996].
- Derivability from finite sets of sequents (hypothesis) is undecidable already in **L**, but it is still enumerable (semi-decidable), as the derivation is finite.

# Algorithmic Problems

- Theoremhood in the Lambek calculus (both **L** and **FL**) is algorithmically decidable, via exhaustive cut-free proof search.
  - More precisely, it is NP-complete for **L** [Pentus 2006] and PSPACE-complete for **FL** [Kanovich 1994; Kanazawa 1996].
- Derivability from finite sets of sequents (hypothesis) is undecidable already in **L**, but it is still enumerable (semi-decidable), as the derivation is finite.
  - When deriving from hypotheses, one cannot completely get rid of cuts.

# Algorithmic Problems

- Theoremhood in the Lambek calculus (both **L** and **FL**) is algorithmically decidable, via exhaustive cut-free proof search.
  - More precisely, it is NP-complete for **L** [Pentus 2006] and PSPACE-complete for **FL** [Kanovich 1994; Kanazawa 1996].
- Derivability from finite sets of sequents (hypothesis) is undecidable already in **L**, but it is still enumerable (semi-decidable), as the derivation is finite.
  - When deriving from hypotheses, one cannot completely get rid of cuts.
  - Derivability from hypotheses in **L** is equivalent to rewriting system reachability / Turing machine halting, in other words, it is $\Sigma^0_1$-complete.

# Algorithmic Problems

- Theoremhood in the Lambek calculus (both **L** and **FL**) is algorithmically decidable, via exhaustive cut-free proof search.
  - More precisely, it is NP-complete for **L** [Pentus 2006] and PSPACE-complete for **FL** [Kanovich 1994; Kanazawa 1996].
- Derivability from finite sets of sequents (hypothesis) is undecidable already in **L**, but it is still enumerable (semi-decidable), as the derivation is finite.
  - When deriving from hypotheses, one cannot completely get rid of cuts.
  - Derivability from hypotheses in **L** is equivalent to rewriting system reachability / Turing machine halting, in other words, it is $\Sigma^0_1$-complete.
- For the infinitary system $\mathbf{ACT}^\omega$, even enumerability is not guaranteed, since here proofs can be infinite objects.

# Algorithmic Problems

- Kleene star can be added to logics extending **FL** with structural rules of contraction and exchange: $\mathbf{FL}_{\mathrm{e}}$, $\mathbf{FL}_{\mathrm{c}}$, $\mathbf{FL}_{\mathrm{ec}}$.

# Algorithmic Problems

- Kleene star can be added to logics extending **FL** with structural rules of contraction and exchange: $\mathbf{FL}_{\mathrm{e}}$, $\mathbf{FL}_{\mathrm{c}}$, $\mathbf{FL}_{\mathrm{ec}}$.
  - In the presence of weakening, $A^*$ trivialises to 1.

# Algorithmic Problems

- Kleene star can be added to logics extending **FL** with structural rules of contraction and exchange: $\mathbf{FL}_{\mathrm{e}}$, $\mathbf{FL}_{\mathrm{c}}$, $\mathbf{FL}_{\mathrm{ec}}$.
  - In the presence of weakening, $A^*$ trivialises to 1.
- In $\mathbf{FL}_{\mathrm{c}}$, contraction appears in the form of 'multi-contraction'

$$\frac{\Gamma, \Pi, \Pi, \Delta \vdash C}{\Gamma, \Pi, \Delta \vdash C}\ \mathrm{MC}$$

(for the sake of cut eliminaton).

# Algorithmic Problems

- Kleene star can be added to logics extending **FL** with structural rules of contraction and exchange: $\mathbf{FL}_{\mathrm{e}}$, $\mathbf{FL}_{\mathrm{c}}$, $\mathbf{FL}_{\mathrm{ec}}$.
  - In the presence of weakening, $A^*$ trivialises to 1.
- In $\mathbf{FL}_{\mathrm{c}}$, contraction appears in the form of 'multi-contraction'

$$\frac{\Gamma, \Pi, \Pi, \Delta \vdash C}{\Gamma, \Pi, \Delta \vdash C}\ \mathrm{MC}$$

  (for the sake of cut eliminaton).
- Even without Kleene star, contraction is a dangerous rule from the point of view of complexity.

# Algorithmic Problems

- Kleene star can be added to logics extending **FL** with structural rules of contraction and exchange: $\mathbf{FL}_{\mathrm{e}}$, $\mathbf{FL}_{\mathrm{c}}$, $\mathbf{FL}_{\mathrm{ec}}$.
  - In the presence of weakening, $A^*$ trivialises to 1.
- In $\mathbf{FL}_{\mathrm{c}}$, contraction appears in the form of 'multi-contraction'

$$\frac{\Gamma, \Pi, \Pi, \Delta \vdash C}{\Gamma, \Pi, \Delta \vdash C}\ \mathrm{MC}$$

  (for the sake of cut eliminaton).
- Even without Kleene star, contraction is a dangerous rule from the point of view of complexity.
- Indeed, in proof search the number of copies of formulae introduced by contraction is uncontrolled.

# Algorithmic Problems

- Kleene star can be added to logics extending **FL** with structural rules of contraction and exchange: $\mathbf{FL}_{\mathrm{e}}$, $\mathbf{FL}_{\mathrm{c}}$, $\mathbf{FL}_{\mathrm{ec}}$.
  - In the presence of weakening, $A^*$ trivialises to 1.
- In $\mathbf{FL}_{\mathrm{c}}$, contraction appears in the form of 'multi-contraction'

$$\frac{\Gamma, \Pi, \Pi, \Delta \vdash C}{\Gamma, \Pi, \Delta \vdash C}\ \mathrm{MC}$$

  (for the sake of cut eliminaton).
- Even without Kleene star, contraction is a dangerous rule from the point of view of complexity.
- Indeed, in proof search the number of copies of formulae introduced by contraction is uncontrolled.
- Yet, $\mathbf{FL}_{\mathrm{ec}}$ is decidable [Kripke 1959], though not primitive recursive.

# Algorithmic Problems

- Kleene star can be added to logics extending **FL** with structural rules of contraction and exchange: $\mathbf{FL}_{\mathrm{e}}$, $\mathbf{FL}_{\mathrm{c}}$, $\mathbf{FL}_{\mathrm{ec}}$.
  - In the presence of weakening, $A^*$ trivialises to 1.
- In $\mathbf{FL}_{\mathrm{c}}$, contraction appears in the form of 'multi-contraction'

$$\frac{\Gamma, \Pi, \Pi, \Delta \vdash C}{\Gamma, \Pi, \Delta \vdash C}\ \mathrm{MC}$$

  (for the sake of cut eliminaton).
- Even without Kleene star, contraction is a dangerous rule from the point of view of complexity.
- Indeed, in proof search the number of copies of formulae introduced by contraction is uncontrolled.
- Yet, $\mathbf{FL}_{\mathrm{ec}}$ is decidable [Kripke 1959], though not primitive recursive.
- $\mathbf{FL}_{\mathrm{c}}$ is undecidable [Chvalovský, Horčík 2016].

# Algorithmic Problems

- For $\mathbf{ACT}^{\omega}$ and its commutative version $\mathbf{ACT}_{\mathrm{e}}^{\omega}$, the theoremhood problem is undecidable, not enumerable, but rather **co-enumerable**.

# Algorithmic Problems

- For $\mathbf{ACT}^{\omega}$ and its commutative version $\mathbf{ACT}_{\mathrm{e}}^{\omega}$, the theoremhood problem is undecidable, not enumerable, but rather **co-enumerable**.
- This means that, whenever a sequent is not provable, this can be witnessed by a finite certificate.

# Algorithmic Problems

- For $\mathbf{ACT}^{\omega}$ and its commutative version $\mathbf{ACT}_{\mathrm{e}}^{\omega}$, the theoremhood problem is undecidable, not enumerable, but rather **co-enumerable**.
- This means that, whenever a sequent is not provable, this can be witnessed by a finite certificate.
- This is dual to the usual situation of enumerable logics (where the certificate is a finite proof).

# Algorithmic Problems

- For $\mathbf{ACT}^{\omega}$ and its commutative version $\mathbf{ACT}_{\mathrm{e}}^{\omega}$, the theoremhood problem is undecidable, not enumerable, but rather **co-enumerable**.
- This means that, whenever a sequent is not provable, this can be witnessed by a finite certificate.
- This is dual to the usual situation of enumerable logics (where the certificate is a finite proof).
- In order to prove undecidability of $\mathbf{ACT}_{\mathrm{e}}^{\omega}$ [K. 2020], we reduce the **non-halting** (infinite run) problem.

# Algorithmic Problems

- For $\mathbf{ACT}^{\omega}$ and its commutative version $\mathbf{ACT}^{\omega}_{\mathrm{e}}$, the theoremhood problem is undecidable, not enumerable, but rather **co-enumerable**.
- This means that, whenever a sequent is not provable, this can be witnessed by a finite certificate.
- This is dual to the usual situation of enumerable logics (where the certificate is a finite proof).
- In order to prove undecidability of $\mathbf{ACT}^{\omega}_{\mathrm{e}}$ [K. 2020], we reduce the **non-halting** (infinite run) problem.
- As our computation model, we shall use counter (Minsky) machines, following [Lincoln et al. 1992].

# Counter Machines

- A counter machine operates a finite number of **counters** (registers) $\mathtt{a}_1, \ldots, \mathtt{a}_k$, holding natural numbers, and has a finite set of **states** $Q$.

# Counter Machines

- A counter machine operates a finite number of **counters** (registers) $\mathtt{a}_1, \ldots, \mathtt{a}_k$, holding natural numbers, and has a finite set of **states** $Q$.
- On the start, $\mathtt{a}_1$ holds the input value, all other counters are zero, and the machine is in its initial state $q_S$.

# Counter Machines

- A counter machine operates a finite number of **counters** (registers) $\mathtt{a}_1, \ldots, \mathtt{a}_k$, holding natural numbers, and has a finite set of **states** $Q$.
- On the start, $\mathtt{a}_1$ holds the input value, all other counters are zero, and the machine is in its initial state $q_S$.
- Commands (instructions):

| | |
|---|---|
| $\textsc{inc}(p, \mathtt{a}_i, q)$ | $p \to q; \quad \mathtt{a}_i := \mathtt{a}_i + 1;$ |
| $\textsc{jzdec}(p, \mathtt{a}_i, q_0, q)$ | if $\mathtt{a}_i = 0$ then $\{ p \to q_0; \}$<br>else $\{ p \to q; \quad \mathtt{a}_i := \mathtt{a}_i - 1; \}$ |

# Counter Machines

- A counter machine operates a finite number of **counters** (registers) $\mathtt{a}_1, \ldots, \mathtt{a}_k$, holding natural numbers, and has a finite set of **states** $Q$.
- On the start, $\mathtt{a}_1$ holds the input value, all other counters are zero, and the machine is in its initial state $q_S$.
- Commands (instructions):

| | |
|---|---|
| $\textsc{inc}(p, \mathtt{a}_i, q)$ | $p \to q; \quad \mathtt{a}_i := \mathtt{a}_i + 1;$ |
| $\textsc{jzdec}(p, \mathtt{a}_i, q_0, q)$ | if $\mathtt{a}_i = 0$ then $\{ p \to q_0; \}$<br>else $\{ p \to q; \quad \mathtt{a}_i := \mathtt{a}_i - 1; \}$ |

- Determinicity: $\leq 1$ command for each $p$.

# Counter Machines

- A counter machine operates a finite number of **counters** (registers) $\mathtt{a}_1, \ldots, \mathtt{a}_k$, holding natural numbers, and has a finite set of **states** $Q$.
- On the start, $\mathtt{a}_1$ holds the input value, all other counters are zero, and the machine is in its initial state $q_S$.
- Commands (instructions):

| | |
|---|---|
| $\textsc{inc}(p, \mathtt{a}_i, q)$ | $p \to q; \quad \mathtt{a}_i := \mathtt{a}_i + 1;$ |
| $\textsc{jzdec}(p, \mathtt{a}_i, q_0, q)$ | if $\mathtt{a}_i = 0$ then $\{ p \to q_0; \}$<br>else $\{ p \to q; \quad \mathtt{a}_i := \mathtt{a}_i - 1; \}$ |

- Determinicity: $\leq 1$ command for each $p$.
- $k = 3$ is sufficient for a Turing-complete computational model: three-counter machines compute all computable functions.

# Counter Machines

- A counter machine operates a finite number of **counters** (registers) $\mathtt{a}_1, \ldots, \mathtt{a}_k$, holding natural numbers, and has a finite set of **states** $Q$.
- On the start, $\mathtt{a}_1$ holds the input value, all other counters are zero, and the machine is in its initial state $q_S$.
- Commands (instructions):

| | |
|---|---|
| $\textsc{inc}(p, \mathtt{a}_i, q)$ | $p \to q; \quad \mathtt{a}_i := \mathtt{a}_i + 1;$ |
| $\textsc{jzdec}(p, \mathtt{a}_i, q_0, q)$ | if $\mathtt{a}_i = 0$ then $\{ p \to q_0; \}$<br>else $\{ p \to q; \quad \mathtt{a}_i := \mathtt{a}_i - 1; \}$ |

- Determinicity: $\leq 1$ command for each $p$.
- $k = 3$ is sufficient for a Turing-complete computational model: three-counter machines compute all computable functions.
- **Non-halting:** given a three-counter machine, decide whether it runs forever (e.g., on zero input).

# Encoding Counter Machines in $\mathbf{ACT}_{\mathrm{e}}^{\omega}$

- Each **configuration** of a three-counter machine is encoded by the following sequence of variables:

$$q, \underbrace{\mathtt{a}_1, \ldots, \mathtt{a}_1}_{n_1}, \underbrace{\mathtt{a}_2, \ldots, \mathtt{a}_2}_{n_2}, \underbrace{\mathtt{a}_3, \ldots, \mathtt{a}_3}_{n_3}$$

(or, briefly: $q, \mathtt{a}_1^{n_1}, \mathtt{a}_2^{n_2}, \mathtt{a}_3^{n_3}$).

# Encoding Counter Machines in $\mathbf{ACT}_{\mathrm{e}}^{\omega}$

- Each **configuration** of a three-counter machine is encoded by the following sequence of variables:

$$q, \underbrace{\mathtt{a}_1, \ldots, \mathtt{a}_1}_{n_1}, \underbrace{\mathtt{a}_2, \ldots, \mathtt{a}_2}_{n_2}, \underbrace{\mathtt{a}_3, \ldots, \mathtt{a}_3}_{n_3}$$

(or, briefly: $q, \mathtt{a}_1^{n_1}, \mathtt{a}_2^{n_2}, \mathtt{a}_3^{n_3}$).

  - In the absence of weakening and contraction, the numbers of occurrences are preserved. The order does not matter.

# Encoding Counter Machines in $\mathbf{ACT}_{\mathrm{e}}^{\omega}$

- Each **configuration** of a three-counter machine is encoded by the following sequence of variables:

$$q, \underbrace{\mathtt{a}_1, \ldots, \mathtt{a}_1}_{n_1}, \underbrace{\mathtt{a}_2, \ldots, \mathtt{a}_2}_{n_2}, \underbrace{\mathtt{a}_3, \ldots, \mathtt{a}_3}_{n_3}$$

  (or, briefly: $q, \mathtt{a}_1^{n_1}, \mathtt{a}_2^{n_2}, \mathtt{a}_3^{n_3}$).
  - In the absence of weakening and contraction, the numbers of occurrences are preserved. The order does not matter.
- We also introduce 'pseudo-states' $z_i$ for zero-checks on $\mathtt{a}_i$.

# Encoding Counter Machines in $\mathbf{ACT}_{\mathrm{e}}^{\omega}$

- Each **configuration** of a three-counter machine is encoded by the following sequence of variables:

$$q, \underbrace{\mathtt{a}_1, \ldots, \mathtt{a}_1}_{n_1}, \underbrace{\mathtt{a}_2, \ldots, \mathtt{a}_2}_{n_2}, \underbrace{\mathtt{a}_3, \ldots, \mathtt{a}_3}_{n_3}$$

  (or, briefly: $q, \mathtt{a}_1^{n_1}, \mathtt{a}_2^{n_2}, \mathtt{a}_3^{n_3}$).
  - In the absence of weakening and contraction, the numbers of occurrences are preserved. The order does not matter.
- We also introduce 'pseudo-states' $z_i$ for zero-checks on $\mathtt{a}_i$.
- The following formula is a regular expression for an arbitrary configuration (where in the pseudo-state $z_i$ the corresponding $\mathtt{a}_i$ should be zero):

$$D = \big((q_1 \vee \ldots \vee q_m) \otimes \mathtt{a}_1^* \otimes \mathtt{a}_2^* \otimes \mathtt{a}_3^*\big) \\ \vee (z_1 \otimes \mathtt{a}_2^* \otimes \mathtt{a}_3^*) \vee (z_2 \otimes \mathtt{a}_1^* \otimes \mathtt{a}_3^*) \vee (z_3 \otimes \mathtt{a}_1^* \otimes \mathtt{a}_2^*).$$

# Encoding Counter Machines in $\mathbf{ACT}_{\mathrm{e}}^{\omega}$

- For each instruction $I$ we introduce the corresponding formula $A_I$:

$$A_{\mathrm{INC}(p,\mathtt{a}_i,q)} = p \backslash (q \otimes \mathtt{a}_i)$$

$$A_{\mathrm{JZDEC}(p,\mathtt{a}_i,q_0,q)} = \big(p \backslash (q_0 \vee z_i)\big) \wedge \big((p \otimes \mathtt{a}_i) \backslash q\big)$$

# Encoding Counter Machines in $\mathbf{ACT}_{\mathrm{e}}^{\omega}$

- For each instruction $I$ we introduce the corresponding formula $A_I$:

$$A_{\mathrm{INC}(p,\mathtt{a}_i,q)} = p \setminus (q \otimes \mathtt{a}_i)$$
$$A_{\mathrm{JZDEC}(p,\mathtt{a}_i,q_0,q)} = \big(p \setminus (q_0 \vee z_i)\big) \wedge \big((p \otimes \mathtt{a}_i) \setminus q\big)$$

- $E = \bigwedge_I A_I \wedge \bigwedge_{i=1}^{3} (z_i \setminus z_i)$ is the formula encoding the machine.

# Encoding Counter Machines in $\mathbf{ACT}_{\mathrm{e}}^{\omega}$

- For each instruction $I$ we introduce the corresponding formula $A_I$:

$$A_{\mathrm{INC}(p,\mathtt{a}_i,q)} = p \setminus (q \otimes \mathtt{a}_i)$$
$$A_{\mathrm{JZDEC}(p,\mathtt{a}_i,q_0,q)} = \big(p \setminus (q_0 \vee z_i)\big) \wedge \big((p \otimes \mathtt{a}_i) \setminus q\big)$$

- $E = \bigwedge_I A_I \wedge \bigwedge_{i=1}^{3} (z_i \setminus z_i)$ is the formula encoding the machine.

**Lemma**

*The machine runs infinitely if and only if the following sequent is provable in* $\mathbf{ACT}_{\mathrm{e}}^{\omega}$*:*

$$E^*, q_S \vdash D.$$

# Encoding Counter Machines in $\mathbf{ACT}_{\mathrm{e}}^{\omega}$

- For each instruction $I$ we introduce the corresponding formula $A_I$:

$$A_{\mathrm{INC}(p,\mathtt{a}_i,q)} = p \backslash (q \otimes \mathtt{a}_i)$$
$$A_{\mathrm{JZDEC}(p,\mathtt{a}_i,q_0,q)} = \big(p \backslash (q_0 \vee z_i)\big) \wedge \big((p \otimes \mathtt{a}_i) \backslash q\big)$$

- $E = \bigwedge_I A_I \wedge \bigwedge_{i=1}^{3} (z_i \backslash z_i)$ is the formula encoding the machine.

## Lemma

*The machine runs infinitely, starting from* $q, \mathtt{a}_1^{n_1}, \mathtt{a}_2^{n_2}, \mathtt{a}_3^{n_3}$, *if and only if the following sequent is provable in* $\mathbf{ACT}_{\mathrm{e}}^{\omega}$*:*

$$E^*, q, \mathtt{a}_1^{n_1}, \mathtt{a}_2^{n_2}, \mathtt{a}_3^{n_3} \vdash D.$$

# Encoding Counter Machines in $\mathbf{ACT}_{\mathrm{e}}^{\omega}$

- For each instruction $I$ we introduce the corresponding formula $A_I$:

$$A_{\mathrm{INC}(p,\mathtt{a}_i,q)} = p \setminus (q \otimes \mathtt{a}_i)$$
$$A_{\mathrm{JZDEC}(p,\mathtt{a}_i,q_0,q)} = \big(p \setminus (q_0 \vee z_i)\big) \wedge \big((p \otimes \mathtt{a}_i) \setminus q\big)$$

- $E = \bigwedge_I A_I \wedge \bigwedge_{i=1}^{3} (z_i \setminus z_i)$ is the formula encoding the machine.

## Lemma

*The machine runs infinitely, starting from* $q, \mathtt{a}_1^{n_1}, \mathtt{a}_2^{n_2}, \mathtt{a}_3^{n_3}$, *if and only if the following sequent is provable in* $\mathbf{ACT}_{\mathrm{e}}^{\omega}$*:*

$$E^*, q, \mathtt{a}_1^{n_1}, \mathtt{a}_2^{n_2}, \mathtt{a}_3^{n_3} \vdash D.$$

- Here $E^*$, by the $*\mathsf{L}_\omega$ rule, instantiates $E^n$ for **any** $n$.

# Encoding Counter Machines in $\mathbf{ACT}_{\mathrm{e}}^{\omega}$

- For each instruction $I$ we introduce the corresponding formula $A_I$:

$$A_{\mathrm{INC}(p,\mathtt{a}_i,q)} = p \setminus (q \otimes \mathtt{a}_i)$$
$$A_{\mathrm{JZDEC}(p,\mathtt{a}_i,q_0,q)} = \big(p \setminus (q_0 \vee z_i)\big) \wedge \big((p \otimes \mathtt{a}_i) \setminus q\big)$$

- $E = \bigwedge_I A_I \wedge \bigwedge_{i=1}^{3} (z_i \setminus z_i)$ is the formula encoding the machine.

## Lemma

*The machine runs infinitely, starting from $q, \mathtt{a}_1^{n_1}, \mathtt{a}_2^{n_2}, \mathtt{a}_3^{n_3}$, if and only if the following sequent is provable in $\mathbf{ACT}_{\mathrm{e}}^{\omega}$:*

$$E^*, q, \mathtt{a}_1^{n_1}, \mathtt{a}_2^{n_2}, \mathtt{a}_3^{n_3} \vdash D.$$

- Here $E^*$, by the $*\mathsf{L}_\omega$ rule, instantiates $E^n$ for **any** $n$.
- The "if" direction (from proof to computation) is not so straightforward and requires some infinitary proof analysis.

# Encoding Counter Machines in $\mathbf{ACT}_{\mathrm{e}}^{\omega}$

- The argument presented above shows that deciding theoremhood in $\mathbf{ACT}_{\mathrm{e}}^{\omega}$ is at least as hard as deciding non-halting for counter machines.

# Encoding Counter Machines in $\mathbf{ACT}_{\mathrm{e}}^{\omega}$

- The argument presented above shows that deciding theoremhood in $\mathbf{ACT}_{\mathrm{e}}^{\omega}$ is at least as hard as deciding non-halting for counter machines.
- The latter is dual to halting, thus undecidable, thus $\mathbf{ACT}_{\mathrm{e}}^{\omega}$ is also undecidable.

# Encoding Counter Machines in $\mathbf{ACT}_{\mathrm{e}}^{\omega}$

- The argument presented above shows that deciding theoremhood in $\mathbf{ACT}_{\mathrm{e}}^{\omega}$ is at least as hard as deciding non-halting for counter machines.
- The latter is dual to halting, thus undecidable, thus $\mathbf{ACT}_{\mathrm{e}}^{\omega}$ is also undecidable.
- The exact complexity class for non-halting is called $\Pi_1^0$, dual to $\Sigma_1^0$ for halting.

# Encoding Counter Machines in $\mathbf{ACT}_{\mathrm{e}}^{\omega}$

- The argument presented above shows that deciding theoremhood in $\mathbf{ACT}_{\mathrm{e}}^{\omega}$ is at least as hard as deciding non-halting for counter machines.
- The latter is dual to halting, thus undecidable, thus $\mathbf{ACT}_{\mathrm{e}}^{\omega}$ is also undecidable.
- The exact complexity class for non-halting is called $\Pi_1^0$, dual to $\Sigma_1^0$ for halting.
  - $\Sigma_1^0$ / $\Pi_1^0$ means one unrestricted $\exists$ / $\forall$ quantifier over natural numbers.

# Encoding Counter Machines in $\mathbf{ACT}_{\mathrm{e}}^{\omega}$

- The argument presented above shows that deciding theoremhood in $\mathbf{ACT}_{\mathrm{e}}^{\omega}$ is at least as hard as deciding non-halting for counter machines.
- The latter is dual to halting, thus undecidable, thus $\mathbf{ACT}_{\mathrm{e}}^{\omega}$ is also undecidable.
- The exact complexity class for non-halting is called $\Pi_1^0$, dual to $\Sigma_1^0$ for halting.
  - $\Sigma_1^0$ / $\Pi_1^0$ means one unrestricted $\exists$ / $\forall$ quantifier over natural numbers.
- $\mathbf{ACT}_{\mathrm{e}}^{\omega}$ is therefore **$\Pi_1^0$-hard,** but we still lack an upper bound (maybe it is even harder?).

# Non-Well-Founded Proofs

- An alternative formulation of $\mathbf{ACT}^{\omega}$ and $\mathbf{ACT}_{\mathrm{e}}^{\omega}$ is the one with **non-well-founded proofs** [Das, Pous 2018].

# Non-Well-Founded Proofs

- An alternative formulation of $\mathbf{ACT}^{\omega}$ and $\mathbf{ACT}^{\omega}_{\mathrm{e}}$ is the one with **non-well-founded proofs** [Das, Pous 2018].
- In this formulation, denoted by $\mathbf{ACT}^{\infty}_{(\mathrm{e})}$, the rules for Kleene star are replaced by the following ones:

$$\frac{\Gamma, \Delta \vdash C \quad \Gamma, A, A^*, \Delta \vdash C}{\Gamma, A^*, \Delta \vdash C}\ {*}\mathsf{L} \qquad \frac{}{\vdash A^*}\ {*}\mathsf{R}_0 \qquad \frac{\Pi \vdash A \quad \Delta \vdash A^*}{\Pi, \Delta \vdash A^*}\ {*}\mathsf{R},$$

where $\Pi$ is non-empty

# Non-Well-Founded Proofs

- An alternative formulation of $\mathbf{ACT}^{\omega}$ and $\mathbf{ACT}^{\omega}_{\mathsf{e}}$ is the one with **non-well-founded proofs** [Das, Pous 2018].
- In this formulation, denoted by $\mathbf{ACT}^{\infty}_{(\mathsf{e})}$, the rules for Kleene star are replaced by the following ones:

$$\frac{\Gamma, \Delta \vdash C \quad \Gamma, A, A^*, \Delta \vdash C}{\Gamma, A^*, \Delta \vdash C}\ {*\mathsf{L}} \qquad \frac{}{\vdash A^*}\ {*\mathsf{R}_0} \qquad \frac{\Pi \vdash A \quad \Delta \vdash A^*}{\Pi, \Delta \vdash A^*}\ {*\mathsf{R}},$$

where $\Pi$ is non-empty

- Now all rules are finitary, but we allow **infinite paths** from the root upwards the tree.

# Non-Well-Founded Proofs

- An alternative formulation of $\mathbf{ACT}^{\omega}$ and $\mathbf{ACT}^{\omega}_{\mathrm{e}}$ is the one with **non-well-founded proofs** [Das, Pous 2018].
- In this formulation, denoted by $\mathbf{ACT}^{\infty}_{(\mathrm{e})}$, the rules for Kleene star are replaced by the following ones:

$$\frac{\Gamma, \Delta \vdash C \quad \Gamma, A, A^*, \Delta \vdash C}{\Gamma, A^*, \Delta \vdash C}\ {*\mathsf{L}} \qquad \frac{}{\vdash A^*}\ {*\mathsf{R}_0} \qquad \frac{\Pi \vdash A \quad \Delta \vdash A^*}{\Pi, \Delta \vdash A^*}\ {*\mathsf{R}},$$

where $\Pi$ is non-empty

- Now all rules are finitary, but we allow **infinite paths** from the root upwards the tree.
- In the presence of Cut, unrestricted usage of such constructions immediately yields absurd:

$$\dfrac{\dfrac{p \vdash p \qquad \dfrac{\vdots}{p \vdash q}}{p \vdash q}\ \mathsf{Cut} \qquad p \vdash p}{p \vdash q}\ \mathsf{Cut}$$

# Non-Well-Founded Proofs

- In the cut-free case, however, no extra conditions are necessary for correctness of non-well-founded proofs!

# Non-Well-Founded Proofs

- In the cut-free case, however, no extra conditions are necessary for correctness of non-well-founded proofs!
  - The “$\Pi$ is non-empty” constraint on $*R$ is important.

# Non-Well-Founded Proofs

- In the cut-free case, however, no extra conditions are necessary for correctness of non-well-founded proofs!
  - The "Π is non-empty" constraint on ∗R is important.

**Theorem (Das, Pous 2018)**

$\mathbf{ACT}^{\omega}_{(\mathrm{e})}$ *and* $\mathbf{ACT}^{\infty}_{(\mathrm{e})}$ *(without* Cut*) prove the same set of sequents.*

# Non-Well-Founded Proofs

- In the cut-free case, however, no extra conditions are necessary for correctness of non-well-founded proofs!
  - The "$\Pi$ is non-empty" constraint on $*\mathrm{R}$ is important.

**Theorem (Das, Pous 2018)**

$\mathbf{ACT}^{\omega}_{(\mathrm{e})}$ *and* $\mathbf{ACT}^{\infty}_{(\mathrm{e})}$ *(without* $\mathsf{Cut}$*) prove the same set of sequents.*

**Lemma**

*If* $\mathbf{ACT}^{\infty}_{(\mathrm{e})}$ *proves* $\Gamma, A^*, \Delta \vdash C$*, then it proves* $\Gamma, A^n, \Delta \vdash C$ *for any* $n$*.*

# Non-Well-Founded Proofs

- In the cut-free case, however, no extra conditions are necessary for correctness of non-well-founded proofs!
  - The "$\Pi$ is non-empty" constraint on $*\mathrm{R}$ is important.

## Theorem (Das, Pous 2018)

$\mathbf{ACT}^{\omega}_{(\mathrm{e})}$ *and* $\mathbf{ACT}^{\infty}_{(\mathrm{e})}$ *(without* $\mathsf{Cut}$*) prove the same set of sequents.*

## Lemma

*If* $\mathbf{ACT}^{\infty}_{(\mathrm{e})}$ *proves* $\Gamma, A^*, \Delta \vdash C$*, then it proves* $\Gamma, A^n, \Delta \vdash C$ *for any* $n$*.*

- This is the key lemma for the harder $\mathbf{ACT}^{\infty}_{(\mathrm{e})} \subseteq \mathbf{ACT}^{\omega}_{(\mathrm{e})}$ direction.

# Non-Well-Founded Proofs

- In the cut-free case, however, no extra conditions are necessary for correctness of non-well-founded proofs!
  - The "$\Pi$ is non-empty" constraint on $*\mathsf{R}$ is important.

## Theorem (Das, Pous 2018)

$\mathbf{ACT}^{\omega}_{(\mathrm{e})}$ *and* $\mathbf{ACT}^{\infty}_{(\mathrm{e})}$ *(without* $\mathsf{Cut}$*) prove the same set of sequents.*

## Lemma

*If* $\mathbf{ACT}^{\infty}_{(\mathrm{e})}$ *proves* $\Gamma, A^*, \Delta \vdash C$*, then it proves* $\Gamma, A^n, \Delta \vdash C$ *for any* $n$*.*

- This is the key lemma for the harder $\mathbf{ACT}^{\infty}_{(\mathrm{e})} \subseteq \mathbf{ACT}^{\omega}_{(\mathrm{e})}$ direction.
- The opposite one is basically modelling $*\mathsf{L}_\omega$ by an non-well-founded proof.

# Non-Well-Founded Proofs

$$\frac{\Gamma, \Delta \vdash C \quad \Gamma, A, \Delta \vdash C \quad \Gamma, A, A, \Delta \vdash C \quad \ldots}{\Gamma, A^*, \Delta \vdash C} *\mathsf{L}_\omega$$

$$\dfrac{\Gamma, \Delta \vdash C \qquad \dfrac{\Gamma, A, \Delta \vdash C \qquad \dfrac{\Gamma, A, A, \Delta \vdash C \qquad \dfrac{\vdots}{\Gamma, A, A, A, A^*, \Delta \vdash C}}{\Gamma, A, A, A^*, \Delta \vdash C} *\mathsf{L}}{\Gamma, A, A^*, \Delta \vdash C} *\mathsf{L}}{\Gamma, A^*, \Delta \vdash C} *\mathsf{L}$$

# Non-Well-Founded Proofs

- The existence of a non-well-founded proof, for a given sequent, is equivalent to the following:

  for any $n$, there exists a proof tree of height $\leq n$ in which every path either reaches an axiom, or has length $n$.

# Non-Well-Founded Proofs

- The existence of a non-well-founded proof, for a given sequent, is equivalent to the following:

  for any $n$, there exists a proof tree of height $\leq n$ in which every path either reaches an axiom, or has length $n$.

  - Proving this equivalence is in fact an application of Kőnig's lemma.

# Non-Well-Founded Proofs

- The existence of a non-well-founded proof, for a given sequent, is equivalent to the following:

  for any $n$, there exists a proof tree of height $\leq n$ in which every path either reaches an axiom, or has length $n$.

  - Proving this equivalence is in fact an application of Kőnig's lemma.
- Let us call this incomplete tree an **$n$-proof.**

# Non-Well-Founded Proofs

- The existence of a non-well-founded proof, for a given sequent, is equivalent to the following:

  for any $n$, there exists a proof tree of height $\leq n$ in which every path either reaches an axiom, or has length $n$.

  - Proving this equivalence is in fact an application of Kőnig's lemma.
- Let us call this incomplete tree an **$n$-proof.**
- Dually, a sequent is **not** provable if and only if one fails to find an $n$-proof for some $n$.

# Non-Well-Founded Proofs

- The existence of a non-well-founded proof, for a given sequent, is equivalent to the following:

  for any $n$, there exists a proof tree of height $\leq n$ in which every path either reaches an axiom, or has length $n$.

  - Proving this equivalence is in fact an application of Kőnig's lemma.
- Let us call this incomplete tree an ***n*-proof.**
- Dually, a sequent is **not** provable if and only if one fails to find an $n$-proof for some $n$.
- For a fixed $n$, there is only a finite number of possible $n$-proofs, so the question whether such exists is algorithmically decidable.

# Non-Well-Founded Proofs

- The existence of a non-well-founded proof, for a given sequent, is equivalent to the following:

  for any $n$, there exists a proof tree of height $\leq n$ in which every path either reaches an axiom, or has length $n$.

  - Proving this equivalence is in fact an application of Kőnig's lemma.
- Let us call this incomplete tree an ***n*-proof.**
- Dually, a sequent is **not** provable if and only if one fails to find an $n$-proof for some $n$.
- For a fixed $n$, there is only a finite number of possible $n$-proofs, so the question whether such exists is algorithmically decidable.
- Thus, $n$ serves as a certificate that a sequent is not provable, and therefore $\mathbf{ACT}^{\omega}_{(\mathrm{e})}$ is in $\Pi^0_1$ (co-enumerable).

# Non-Well-Founded Proofs

- Non-well-founded proofs are also more natural for modelling infinite computations.

# Non-Well-Founded Proofs

- Non-well-founded proofs are also more natural for modelling infinite computations.
- In fact, each computation step is modelled by a subderivation from one configuration to another.

# Non-Well-Founded Proofs

- Non-well-founded proofs are also more natural for modelling infinite computations.
- In fact, each computation step is modelled by a subderivation from one configuration to another.
- For example, $\textsc{inc}(p, \mathtt{a}_1, q)$ is modelled as follows:

$$\dfrac{p, \mathtt{a}_1^{n_1}, \mathtt{a}_2^{n_2}, \mathtt{a}_3^{n_3} \vdash D \qquad \dfrac{\dfrac{\dfrac{E^*, q, \mathtt{a}_1^{n_1+1}, \mathtt{a}_2^{n_2}, \mathtt{a}_3^{n_3} \vdash D}{E^*, q \otimes \mathtt{a}_1, \mathtt{a}_1^{n_1}, \mathtt{a}_2^{n_2}, \mathtt{a}_3^{n_3} \vdash D}\otimes\mathsf{L} \quad p \vdash p}{E^*, p \setminus (q \otimes \mathtt{a}_1), p, \mathtt{a}_1^{n_1}, \mathtt{a}_2^{n_2}, \mathtt{a}_3^{n_3} \vdash D}\setminus\mathsf{L}}{E, E^*, p, \mathtt{a}_1^{n_1}, \mathtt{a}_2^{n_2}, \mathtt{a}_3^{n_3} \vdash D}\wedge\mathsf{L}}{E^*, p, \mathtt{a}_1^{n_1}, \mathtt{a}_2^{n_2}, \mathtt{a}_3^{n_3} \vdash D}*\mathsf{L}$$

# Non-Well-Founded Proofs

- Stacking these proof fragments one upon another yields a non-well-founded proof of the initial sequent $E^*, q_0 \vdash D$.

# Non-Well-Founded Proofs

- Stacking these proof fragments one upon another yields a non-well-founded proof of the initial sequent $E^*, q_0 \vdash D$.
- For zero-check, $\textsc{jzdec}(p, \mathtt{a}_1, q_0, q_1)$, where the value of $\mathtt{a}_1$ is zero, the derivation is a bit more involved:

$$\dfrac{\dfrac{p, \mathtt{a}_2^{n_2}, \mathtt{a}_3^{n_3} \vdash D \qquad \dfrac{\dfrac{\dfrac{E^*, q_0, \mathtt{a}_2^{n_2}, \mathtt{a}_3^{n_3} \vdash D \quad E^*, z_1, \mathtt{a}_2^{n_2}, \mathtt{a}_3^{n_3} \vdash D}{E^*, q_0 \vee z_1, \mathtt{a}_2^{n_2}, \mathtt{a}_3^{n_3} \vdash D}\vee\mathsf{L} \qquad p \vdash p}{E^*, p \setminus (q_0 \vee z_1), p, \mathtt{a}_2^{n_2}, \mathtt{a}_3^{n_3} \vdash D}\setminus\mathsf{L}}{E, E^*, p, \mathtt{a}_2^{n_2}, \mathtt{a}_3^{n_3} \vdash D}\wedge\mathsf{L}}{E^*, p, \mathtt{a}_2^{n_2}, \mathtt{a}_3^{n_3} \vdash D}*\mathsf{L}$$

# Circular Proofs

- For $E^*, z_1, \mathtt{a}_2^{n_2}, \mathtt{a}_3^{n_3} \vdash D$, we use the following non-well-founded proof, which is in fact a **circular** one:

$$\dfrac{z_1, \mathtt{a}_2^{n_2}, \mathtt{a}_3^{n_3} \vdash D \qquad \dfrac{\dfrac{z_1 \vdash z_1 \quad E^*, z_1, \mathtt{a}_2^{n_2}, \mathtt{a}_3^{n_3} \vdash D}{E^*, z_1 \setminus z_1, z_1, \mathtt{a}_2^{n_2}, \mathtt{a}_3^{n_3} \vdash D}\,\setminus\mathsf{L}}{E, E^*, z_1, \mathtt{a}_2^{n_2}, \mathtt{a}_3^{n_3} \vdash D}\,\wedge\mathsf{L}}{E^*, z_1, \mathtt{a}_2^{n_2}, \mathtt{a}_3^{n_3} \vdash D}\,*\mathsf{L}$$

# Circular Proofs

- For $E^*, z_1, \mathtt{a}_2^{n_2}, \mathtt{a}_3^{n_3} \vdash D$, we use the following non-well-founded proof, which is in fact a **circular** one:

$$\dfrac{z_1, \mathtt{a}_2^{n_2}, \mathtt{a}_3^{n_3} \vdash D \qquad \dfrac{\dfrac{\dfrac{z_1 \vdash z_1 \quad E^*, z_1, \mathtt{a}_2^{n_2}, \mathtt{a}_3^{n_3} \vdash D}{E^*, z_1 \setminus z_1, z_1, \mathtt{a}_2^{n_2}, \mathtt{a}_3^{n_3} \vdash D}\,\setminus\mathsf{L}}{E, E^*, z_1, \mathtt{a}_2^{n_2}, \mathtt{a}_3^{n_3} \vdash D}\,\wedge\mathsf{L}}{E^*, z_1, \mathtt{a}_2^{n_2}, \mathtt{a}_3^{n_3} \vdash D}\,*\mathsf{L}$$

- Circular proofs are correct (surprise!), as a special case of non-well-founded proofs.

# Circular Proofs

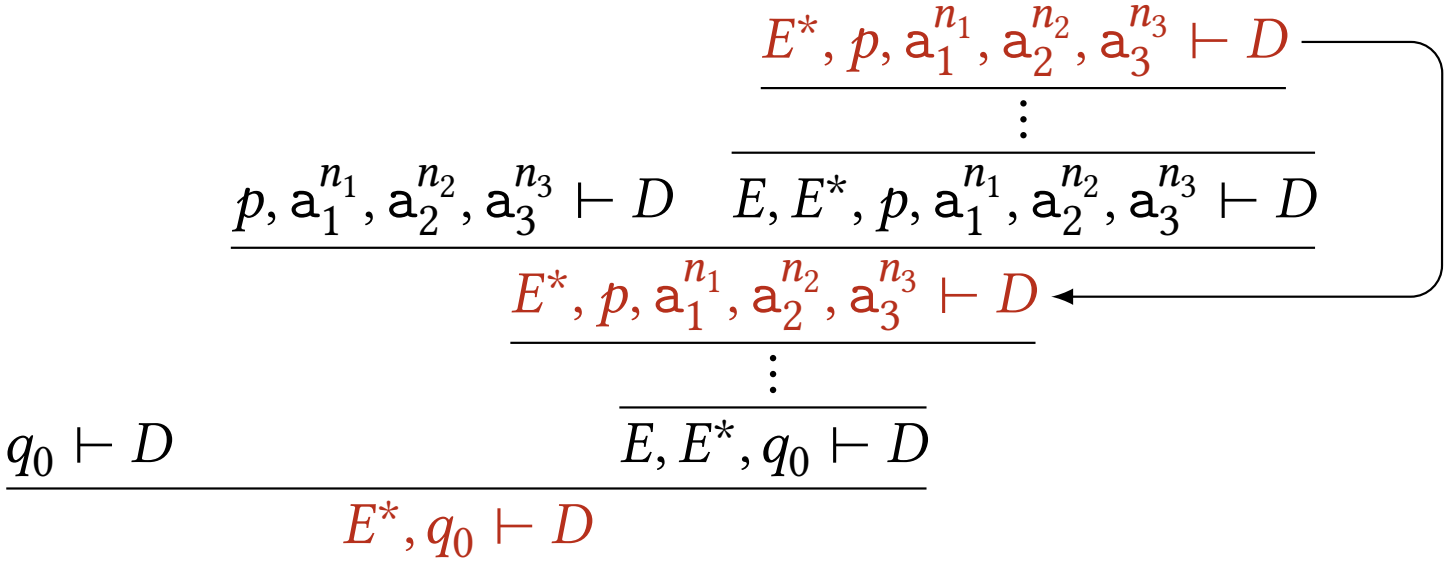
- For $E^*, z_1, \mathtt{a}_2^{n_2}, \mathtt{a}_3^{n_3} \vdash D$, we use the following non-well-founded proof, which is in fact a **circular** one:

$$\dfrac{z_1, \mathtt{a}_2^{n_2}, \mathtt{a}_3^{n_3} \vdash D \qquad \dfrac{\dfrac{\dfrac{z_1 \vdash z_1 \quad E^*, z_1, \mathtt{a}_2^{n_2}, \mathtt{a}_3^{n_3} \vdash D}{E^*, z_1 \setminus z_1, z_1, \mathtt{a}_2^{n_2}, \mathtt{a}_3^{n_3} \vdash D}\,\setminus\mathsf{L}}{E, E^*, z_1, \mathtt{a}_2^{n_2}, \mathtt{a}_3^{n_3} \vdash D}\,\wedge\mathsf{L}}{E^*, z_1, \mathtt{a}_2^{n_2}, \mathtt{a}_3^{n_3} \vdash D}\,*\mathsf{L}$$

- Circular proofs are correct (surprise!), as a special case of non-well-founded proofs.
- In fact, at each iteration of the cycle, the proof makes *progress* by passing through $*\mathsf{L}$.

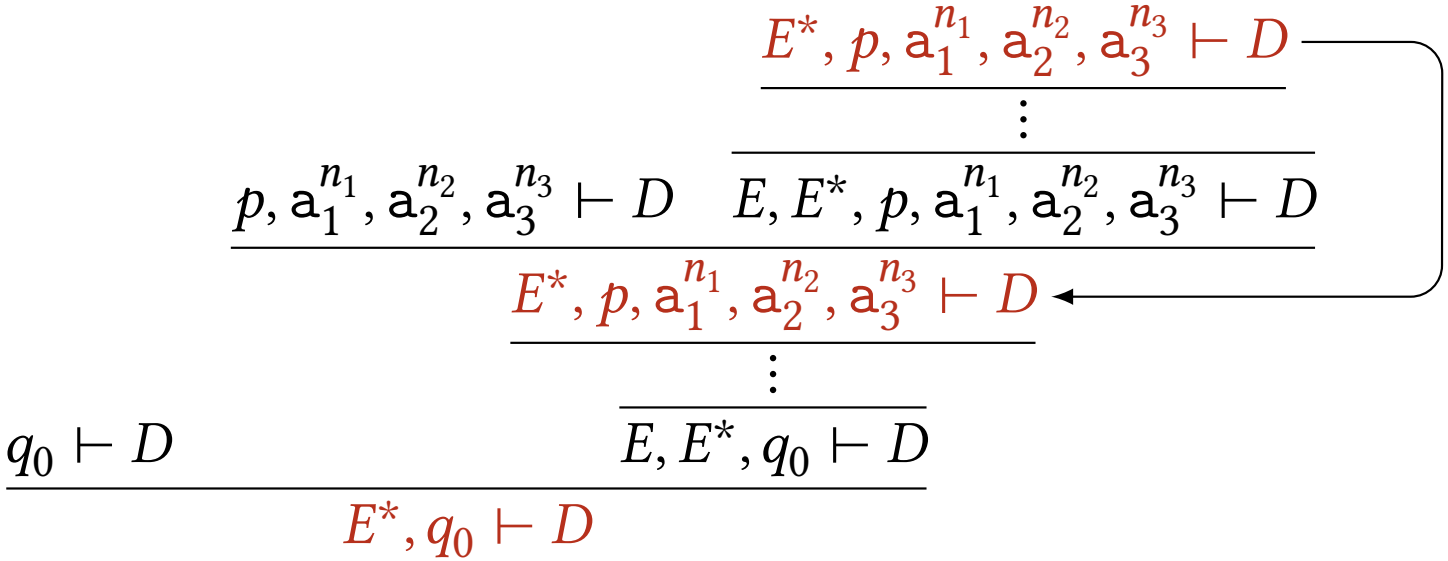
# Circular Proofs

- Circular proofs are capable of modelling circular (periodic) computations on counter machines:



# Circular Proofs

- Circular proofs are capable of modelling circular (periodic) computations on counter machines:




- On the other hand, circular proofs cannot model arbitrary infinitary computations, as they form an enumerable logic.

# Circular Proofs

- In fact, the circular fragment of $\mathbf{ACT}^{\infty}_{(c)}$ is equivalent to (commutative) **action logic** $\mathbf{ACT}_{(c)}$, where $A^*$ has a fixed point (inductive) axiomatisation [Pratt 1991; Kozen 1994]:

$$\frac{\vdash B \quad A, B \vdash B}{A^* \vdash B}\ {*\mathsf{L}_{\mathsf{fix}}} \qquad \frac{}{\vdash A^*}\ {*\mathsf{R}_0} \qquad \frac{\Pi \vdash A \quad \Delta \vdash A^*}{\Pi, \Delta \vdash A^*}\ {*\mathsf{R}}$$

# Circular Proofs

- In fact, the circular fragment of $\mathbf{ACT}^{\infty}_{(c)}$ is equivalent to (commutative) **action logic** $\mathbf{ACT}_{(c)}$, where $A^*$ has a fixed point (inductive) axiomatisation [Pratt 1991; Kozen 1994]:

$$\frac{\vdash B \quad A, B \vdash B}{A^* \vdash B}\ {*\mathsf{L}_{\mathsf{fix}}} \qquad \frac{}{\vdash A^*}\ {*\mathsf{R}_0} \qquad \frac{\Pi \vdash A \quad \Delta \vdash A^*}{\Pi, \Delta \vdash A^*}\ {*\mathsf{R}}$$

- Unfortunately, for these systems no cut-free calculi are known, so we have no way to prove the other direction: from circular proofs to circular computations.

# Circular Proofs

- In fact, the circular fragment of $\mathbf{ACT}^{\infty}_{(c)}$ is equivalent to (commutative) **action logic** $\mathbf{ACT}_{(c)}$, where $A^*$ has a fixed point (inductive) axiomatisation [Pratt 1991; Kozen 1994]:

$$\frac{\vdash B \quad A, B \vdash B}{A^* \vdash B}\ {*\mathsf{L}_{\mathsf{fix}}} \qquad \frac{}{\vdash A^*}\ {*\mathsf{R}_0} \qquad \frac{\Pi \vdash A \quad \Delta \vdash A^*}{\Pi, \Delta \vdash A^*}\ {*\mathsf{R}}$$

- Unfortunately, for these systems no cut-free calculi are known, so we have no way to prove the other direction: from circular proofs to circular computations.
- Undecidability of $\mathbf{ACT}_{\mathsf{e}}$ [K. 2024] is proved by an indirect technique of **inseparability**.

# Circular Proofs

- In fact, the circular fragment of $\mathbf{ACT}^{\infty}_{(c)}$ is equivalent to (commutative) **action logic** $\mathbf{ACT}_{(c)}$, where $A^*$ has a fixed point (inductive) axiomatisation [Pratt 1991; Kozen 1994]:

$$\frac{\vdash B \quad A, B \vdash B}{A^* \vdash B}\ {*\mathsf{L}_{\mathsf{fix}}} \qquad \frac{}{\vdash A^*}\ {*\mathsf{R}_0} \qquad \frac{\Pi \vdash A \quad \Delta \vdash A^*}{\Pi, \Delta \vdash A^*}\ {*\mathsf{R}}$$

- Unfortunately, for these systems no cut-free calculi are known, so we have no way to prove the other direction: from circular proofs to circular computations.
- Undecidability of $\mathbf{ACT}_{\mathsf{e}}$ [K. 2024] is proved by an indirect technique of **inseparability**.
- Any set separating circular computation from halting is undecidable.

# Circular Proofs

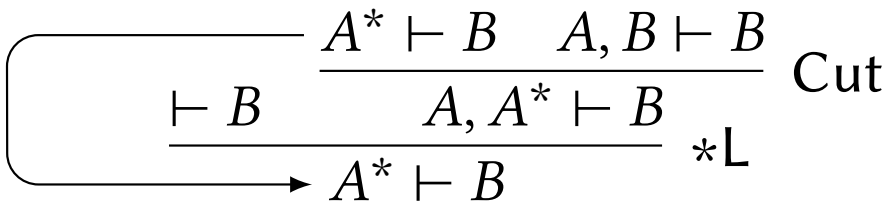
- In fact, the circular fragment of $\mathbf{ACT}^{\infty}_{(c)}$ is equivalent to (commutative) **action logic** $\mathbf{ACT}_{(c)}$, where $A^*$ has a fixed point (inductive) axiomatisation [Pratt 1991; Kozen 1994]:

$$\frac{\vdash B \quad A, B \vdash B}{A^* \vdash B}\ {*\mathsf{L}_{\mathsf{fix}}} \qquad \frac{}{\vdash A^*}\ {*\mathsf{R}_0} \qquad \frac{\Pi \vdash A \quad \Delta \vdash A^*}{\Pi, \Delta \vdash A^*}\ {*\mathsf{R}}$$

- Unfortunately, for these systems no cut-free calculi are known, so we have no way to prove the other direction: from circular proofs to circular computations.
- Undecidability of $\mathbf{ACT}_{\mathsf{e}}$ [K. 2024] is proved by an indirect technique of **inseparability**.
- Any set separating circular computation from halting is undecidable.
- If it is enumerable, then it is $\Sigma^0_1$-complete (effective inseparability).

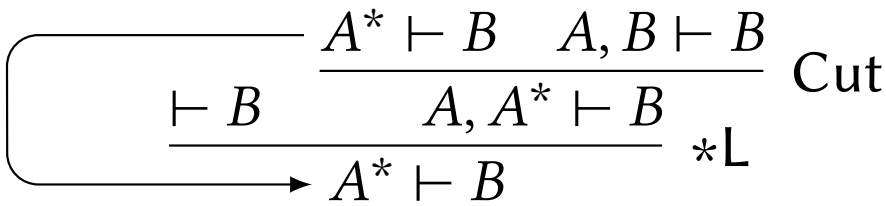
# Circular Proofs and Induction

- Modelling $*\mathsf{L}_{\text{fix}}$ via circular proof:



# Circular Proofs and Induction

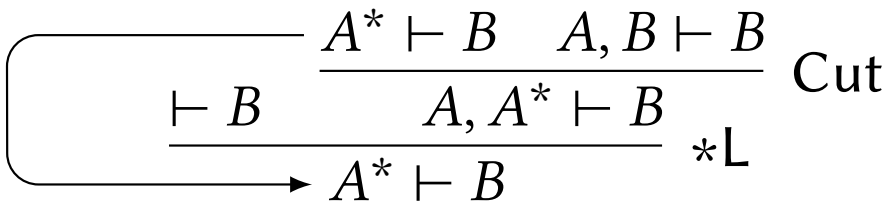
- Modelling $*\mathsf{L}_{\text{fix}}$ via circular proof:




- In circular proofs, in general, we allow cuts. Correctness condition: each cyclic backlink points to the conclusion of a 'validating' $*\mathsf{L}$, and the cycle goes through its $2^{\text{nd}}$ premise.

# Circular Proofs and Induction

- Modelling $*\mathsf{L}_{\text{fix}}$ via circular proof:




- In circular proofs, in general, we allow cuts. Correctness condition: each cyclic backlink points to the conclusion of a 'validating' $*\mathsf{L}$, and the cycle goes through its $2^{\text{nd}}$ premise.
- In order to translate a circular proof into an inductive one, consider one bunch of cycles which backtrack to $\Gamma, A^*, \Delta \vdash C$ under $*\mathsf{L}$.

# Circular Proofs and Induction

- Modelling $*\mathsf{L}_{\text{fix}}$ via circular proof:

$$\dfrac{\vdash B \qquad \dfrac{A^* \vdash B \quad A, B \vdash B}{A, A^* \vdash B}\,\text{Cut}}{A^* \vdash B}\,*\mathsf{L}$$

- In circular proofs, in general, we allow cuts. Correctness condition: each cyclic backlink points to the conclusion of a 'validating' $*\mathsf{L}$, and the cycle goes through its $2^{\text{nd}}$ premise.
- In order to translate a circular proof into an inductive one, consider one bunch of cycles which backtrack to $\Gamma, A^*, \Delta \vdash C$ under $*\mathsf{L}$.
- Let us track the 'active' $A^*$ and denote the sequents where it occurs by $\Gamma_i, A^*, \Delta_i \vdash C_i$, where $i = 1, \ldots, n$.

# Circular Proofs and Induction

- Let $E_i = \Gamma_i \setminus C_i \,/\, \Delta_i$ and let $E = E_1 \wedge \ldots \wedge E_n$.

# Circular Proofs and Induction

- Let $E_i = \Gamma_i \setminus C_i \,/\, \Delta_i$ and let $E = E_1 \wedge \ldots \wedge E_n$.
- In the proof of $\Gamma, A, A^*, \Delta \vdash C$, replace each 'active' $A^*$ with $E$.

# Circular Proofs and Induction

- Let $E_i = \Gamma_i \setminus C_i \,/\, \Delta_i$ and let $E = E_1 \wedge \ldots \wedge E_n$.
- In the proof of $\Gamma, A, A^*, \Delta \vdash C$, replace each 'active' $A^*$ with $E$.
- Now further applications of $*\mathsf{L}$, with 'active' $A^*$, in this proof trivialise: $\Gamma_j, A^*, \Delta_j \vdash C_j$ becomes $\Gamma_j, E, \Delta_j \vdash C_j$, which is provable by choosing $E_j$ from $E$.

# Circular Proofs and Induction

- Let $E_i = \Gamma_i \setminus C_i \,/\, \Delta_i$ and let $E = E_1 \wedge \ldots \wedge E_n$.
- In the proof of $\Gamma, A, A^*, \Delta \vdash C$, replace each 'active' $A^*$ with $E$.
- Now further applications of $*\mathsf{L}$, with 'active' $A^*$, in this proof trivialise: $\Gamma_j, A^*, \Delta_j \vdash C_j$ becomes $\Gamma_j, E, \Delta_j \vdash C_j$, which is provable by choosing $E_j$ from $E$.
- Now for the goal sequent we have $\Gamma, \Delta \vdash C$ and $\Gamma, A, E, \Delta \vdash C$ provable.

# Circular Proofs and Induction

- Let $E_i = \Gamma_i \setminus C_i \,/\, \Delta_i$ and let $E = E_1 \wedge \ldots \wedge E_n$.
- In the proof of $\Gamma, A, A^*, \Delta \vdash C$, replace each 'active' $A^*$ with $E$.
- Now further applications of $*\mathsf{L}$, with 'active' $A^*$, in this proof trivialise: $\Gamma_j, A^*, \Delta_j \vdash C_j$ becomes $\Gamma_j, E, \Delta_j \vdash C_j$, which is provable by choosing $E_j$ from $E$.
- Now for the goal sequent we have $\Gamma, \Delta \vdash C$ and $\Gamma, A, E, \Delta \vdash C$ provable.
- Equivalently for $C' = \Gamma \setminus C \,/\, \Delta$ we have $\vdash C'$ and $A, E \vdash C'$, and via cut with $C' \vdash E$ we have $A, E \vdash E$.

# Circular Proofs and Induction

- Let $E_i = \Gamma_i \setminus C_i \,/\, \Delta_i$ and let $E = E_1 \wedge \ldots \wedge E_n$.
- In the proof of $\Gamma, A, A^*, \Delta \vdash C$, replace each 'active' $A^*$ with $E$.
- Now further applications of $*\mathsf{L}$, with 'active' $A^*$, in this proof trivialise: $\Gamma_j, A^*, \Delta_j \vdash C_j$ becomes $\Gamma_j, E, \Delta_j \vdash C_j$, which is provable by choosing $E_j$ from $E$.
- Now for the goal sequent we have $\Gamma, \Delta \vdash C$ and $\Gamma, A, E, \Delta \vdash C$ provable.
- Equivalently for $C' = \Gamma \setminus C \,/\, \Delta$ we have $\vdash C'$ and $A, E \vdash C'$, and via cut with $C' \vdash E$ we have $A, E \vdash E$.
- Thus, $A^* \vdash E$, hence $\Gamma, A^*, \Delta \vdash C$.

# Context-Free Grammars

- For the non-commutative systems $\mathbf{ACT}^{\omega}$ and $\mathbf{ACT}$, direct encoding of computation is impossible.

# Context-Free Grammars

- For the non-commutative systems $\mathbf{ACT}^{\omega}$ and **ACT**, direct encoding of computation is impossible.
- In order to prove their undecidability, Buszkowski [2007] suggested encoding the **totality problem** for context-free grammars.

# Context-Free Grammars

- For the non-commutative systems $\mathbf{ACT}^{\omega}$ and **ACT**, direct encoding of computation is impossible.
- In order to prove their undecidability, Buszkowski [2007] suggested encoding the **totality problem** for context-free grammars.
- Recall that context-free grammars (cf-grammars) form yet another framework for describing languages.

# Context-Free Grammars

- For the non-commutative systems $\mathbf{ACT}^{\omega}$ and **ACT**, direct encoding of computation is impossible.
- In order to prove their undecidability, Buszkowski [2007] suggested encoding the **totality problem** for context-free grammars.
- Recall that context-free grammars (cf-grammars) form yet another framework for describing languages.
- In a cf-grammar, the sentence is built recursively from a starting meta-symbol $S$ by certain **production rules.**

# Context-Free Grammars

- For the non-commutative systems $\mathbf{ACT}^{\omega}$ and **ACT**, direct encoding of computation is impossible.
- In order to prove their undecidability, Buszkowski [2007] suggested encoding the **totality problem** for context-free grammars.
- Recall that context-free grammars (cf-grammars) form yet another framework for describing languages.
- In a cf-grammar, the sentence is built recursively from a starting meta-symbol $S$ by certain **production rules.**
- Meta-symbols (non-terminals) form an alphabet $N$, while $\Sigma$ is the original alphabet.

# Context-Free Grammars

- For the non-commutative systems $\mathbf{ACT}^{\omega}$ and **ACT**, direct encoding of computation is impossible.
- In order to prove their undecidability, Buszkowski [2007] suggested encoding the **totality problem** for context-free grammars.
- Recall that context-free grammars (cf-grammars) form yet another framework for describing languages.
- In a cf-grammar, the sentence is built recursively from a starting meta-symbol $S$ by certain **production rules.**
- Meta-symbols (non-terminals) form an alphabet $N$, while $\Sigma$ is the original alphabet.
- Each production rule is of the form $A \Rightarrow \alpha$, where $A \in N$, $\alpha \in (N \cup \Sigma)^*$.

# Context-Free Grammars

- Example: *John loves Mary*

$$S \Rightarrow NP\ VP \qquad NP \Rightarrow John$$
$$VP \Rightarrow V\ NP \qquad NP \Rightarrow Mary$$
$$V \Rightarrow loves$$

# Context-Free Grammars

- Example: *John loves Mary*

$$S \Rightarrow NP\ VP \qquad NP \Rightarrow John$$
$$VP \Rightarrow V\ NP \qquad NP \Rightarrow Mary$$
$$V \Rightarrow loves$$

- In Lambek grammar:

$$np, (np \setminus s) / np, np \vdash s.$$

# Context-Free Grammars

- Example: *John loves Mary*

$$S \Rightarrow NP\ VP \qquad NP \Rightarrow John$$
$$VP \Rightarrow V\ NP \qquad NP \Rightarrow Mary$$
$$V \Rightarrow loves$$

- In Lambek grammar:

$$np, (np \setminus s) / np, np \vdash s.$$

**Theorem (Bar-Hillel et al. 1961; Pentus 1992)**

*Lambek grammars generate exactly context-free languages without the empty word.*

# Context-Free Grammars

- Example: *John loves Mary*

$$S \Rightarrow NP\ VP \qquad NP \Rightarrow John$$
$$VP \Rightarrow V\ NP \qquad NP \Rightarrow Mary$$
$$V \Rightarrow loves$$

- In Lambek grammar:

$$np, (np \setminus s) / np, np \vdash s.$$

**Theorem (Bar-Hillel et al. 1961; Pentus 1992)**

*Lambek grammars generate exactly context-free languages without the empty word.*

- We shall need the easier part: from cf-grammars to Lambek grammars.

# Context-Free and Lambek Grammars

- Lambek grammars are **lexicalised:** all concrete language information is stored in the lexicon (type assignment).

# Context-Free and Lambek Grammars

- Lambek grammars are **lexicalised:** all concrete language information is stored in the lexicon (type assignment).
- In contrast, cf-grammars use **global** production rules.

# Context-Free and Lambek Grammars

- Lambek grammars are **lexicalised:** all concrete language information is stored in the lexicon (type assignment).
- In contrast, cf-grammars use **global** production rules.
- A way to lexicalise a cf-grammar is the **Greibach normal form** [Greibach 1965]:

$$A \Rightarrow aBC \qquad A \Rightarrow aB \qquad A \Rightarrow a$$

(here $A, B, C \in N$, $a \in \Sigma$).

# Context-Free and Lambek Grammars

- Lambek grammars are **lexicalised:** all concrete language information is stored in the lexicon (type assignment).
- In contrast, cf-grammars use **global** production rules.
- A way to lexicalise a cf-grammar is the **Greibach normal form** [Greibach 1965]:

$$A \Rightarrow aBC \qquad A \Rightarrow aB \qquad A \Rightarrow a$$

(here $A, B, C \in N$, $a \in \Sigma$).

- Example:

$$S \Rightarrow John\ \ VP \qquad NP \Rightarrow John$$
$$S \Rightarrow Mary\ \ VP \qquad NP \Rightarrow Mary$$
$$VP \Rightarrow loves\ \ NP$$

# Context-Free Grammars and Lambek Grammars

- Greibachisation can lead to significant growth of the grammar size.

# Context-Free Grammars and Lambek Grammars

- Greibachisation can lead to significant growth of the grammar size.
- If a grammar is in Greibach normal form, it is easily translated to a Lambek grammar:

$$\begin{array}{ccc} A \Rightarrow aBC & A \Rightarrow aB & A \Rightarrow a \\ a \rhd A \,/\, (B \otimes C) & a \rhd A \,/\, B & a \rhd A \end{array}$$

# Context-Free Grammars and Lambek Grammars

- Greibachisation can lead to significant growth of the grammar size.
- If a grammar is in Greibach normal form, it is easily translated to a Lambek grammar:

$$\begin{array}{ccc} A \Rightarrow aBC & A \Rightarrow aB & A \Rightarrow a \\ a \rhd A \,/\, (B \otimes C) & a \rhd A \,/\, B & a \rhd A \end{array}$$

- Example:

$$\begin{array}{l} \mathit{John}, \mathit{Mary} \rhd np, \quad s \,/\, vp \\ \mathit{loves} \rhd vp \,/\, np \end{array}$$

# Undecidability of $\mathbf{ACT}^{\omega}$

- A cf-grammar is **total,** if it generates all non-empty words.

# Undecidability of $\mathbf{ACT}^{\omega}$

- A cf-grammar is **total,** if it generates all non-empty words.
- There is a reduction from non-halting of (say, counter) machines to totality of cf-grammars.

# Undecidability of $\mathbf{ACT}^{\omega}$

- A cf-grammar is **total,** if it generates all non-empty words.
- There is a reduction from non-halting of (say, counter) machines to totality of cf-grammars.
  - The idea is based on the fact that, while $\{ww \mid w \in \Sigma^*\}$ is not context-free, its *complement* is.

# Undecidability of $\mathbf{ACT}^{\omega}$

- A cf-grammar is **total,** if it generates all non-empty words.
- There is a reduction from non-halting of (say, counter) machines to totality of cf-grammars.
  - The idea is based on the fact that, while $\{ww \mid w \in \Sigma^*\}$ is not context-free, its *complement* is.
- We transform the cf-grammar to a Lambek grammar, and let

$$F_a = \bigwedge \{A \mid a \rhd A\}; \qquad E = \bigvee \{F_a \mid a \in \Sigma\}.$$

# Undecidability of $\mathbf{ACT}^\omega$

- A cf-grammar is **total,** if it generates all non-empty words.
- There is a reduction from non-halting of (say, counter) machines to totality of cf-grammars.
  - The idea is based on the fact that, while $\{ww \mid w \in \Sigma^*\}$ is not context-free, its *complement* is.
- We transform the cf-grammar to a Lambek grammar, and let

$$F_a = \bigwedge \{A \mid a \rhd A\}; \qquad E = \bigvee \{F_a \mid a \in \Sigma\}.$$

- Now the grammar is total if and only if the following is provable in $\mathbf{ACT}^\omega$:

$$E, E^* \vdash s.$$

# Undecidability of $\mathbf{ACT}^\omega$

- A cf-grammar is **total,** if it generates all non-empty words.
- There is a reduction from non-halting of (say, counter) machines to totality of cf-grammars.
  - The idea is based on the fact that, while $\{ww \mid w \in \Sigma^*\}$ is not context-free, its *complement* is.
- We transform the cf-grammar to a Lambek grammar, and let

$$F_a = \bigwedge \{A \mid a \rhd A\}; \qquad E = \bigvee \{F_a \mid a \in \Sigma\}.$$

- Now the grammar is total if and only if the following is provable in $\mathbf{ACT}^\omega$:

$$E, E^* \vdash s.$$

- This gives undecidability of $\mathbf{ACT}^\omega$ [Buszkowski 2007]; for **ACT**, we again use inseparability [K. 2019].

# Reasoning from Hypotheses

- Now let us consider derivability from **hypotheses,** i.e., extra principles added as 'axioms.'

# Reasoning from Hypotheses

- Now let us consider derivability from **hypotheses,** i.e., extra principles added as 'axioms.'
- For example, we may want $n \vdash n \,/\, n$, meaning that any common noun can act as an adjective (*cat* $\rightarrow$ *cat food*).

# Reasoning from Hypotheses

- Now let us consider derivability from **hypotheses,** i.e., extra principles added as 'axioms.'
- For example, we may want $n \vdash n \,/\, n$, meaning that any common noun can act as an adjective (*cat* $\rightarrow$ *cat food*).
- Deriving from hypotheses, in general, requires the usage of Cut:

$$\dfrac{n \vdash n \,/\, n \qquad \dfrac{n \vdash n \quad n \vdash n}{n \,/\, n, n \vdash n}\,/\mathrm{L}}{n, n \vdash n}\,\mathrm{Cut}$$

# Reasoning from Hypotheses

- Now let us consider derivability from **hypotheses,** i.e., extra principles added as 'axioms.'
- For example, we may want $n \vdash n \,/\, n$, meaning that any common noun can act as an adjective (*cat* $\rightarrow$ *cat food*).
- Deriving from hypotheses, in general, requires the usage of Cut:

$$\dfrac{n \vdash n \,/\, n \qquad \dfrac{n \vdash n \quad n \vdash n}{n \,/\, n, n \vdash n}\,/\mathrm{L}}{n, n \vdash n}\,\mathrm{Cut}$$

- Thus, our finite proof search algorithm does not work, and indeed derivability from finite sets of hypothesis in **L** (and therefore **FL**) is undecidable.

# Reasoning from Hypotheses

- Now let us consider derivability from **hypotheses,** i.e., extra principles added as 'axioms.'
- For example, we may want $n \vdash n \,/\, n$, meaning that any common noun can act as an adjective (*cat* $\rightarrow$ *cat food*).
- Deriving from hypotheses, in general, requires the usage of Cut:

$$\dfrac{n \vdash n \,/\, n \qquad \dfrac{n \vdash n \quad n \vdash n}{n \,/\, n, n \vdash n}\,/\mathrm{L}}{n, n \vdash n}\,\mathrm{Cut}$$

- Thus, our finite proof search algorithm does not work, and indeed derivability from finite sets of hypothesis in **L** (and therefore **FL**) is undecidable.
- This is also valid in the commutative case, but one should take extra care of the zero-checks.

# Reasoning from Hypothesis

- What about derivability from hypotheses in $\mathbf{ACT}^{\omega}_{(\mathrm{c})}$ ?

# Reasoning from Hypothesis

- What about derivability from hypotheses in $\mathbf{ACT}^{\omega}_{(\mathrm{c})}$ ?
- Let us first establish an upper bound.

# Reasoning from Hypothesis

- What about derivability from hypotheses in $\mathbf{ACT}^{\omega}_{(\mathrm{c})}$ ?
- Let us first establish an upper bound.
- A sequent $\Pi \vdash A$ is derivable from a set of hypotheses $\mathscr{H}$ if and only if $\Pi \vdash A$ belongs to **any** set of sequents including $\mathscr{H}$, including all axioms and closed under all rules of $\mathbf{ACT}^{\omega}_{(\mathrm{c})}$.

# Reasoning from Hypothesis

- What about derivability from hypotheses in $\mathbf{ACT}^{\omega}_{(\mathrm{c})}$ ?
- Let us first establish an upper bound.
- A sequent $\Pi \vdash A$ is derivable from a set of hypotheses $\mathscr{H}$ if and only if $\Pi \vdash A$ belongs to **any** set of sequents including $\mathscr{H}$, including all axioms and closed under all rules of $\mathbf{ACT}^{\omega}_{(\mathrm{c})}$.
- This condition has one quantifier over **sets** of natural numbers (**second-order** arithmetical quantifier), followed by a property expressible in **first-order arithmetic.**

# Reasoning from Hypothesis

- What about derivability from hypotheses in $\mathbf{ACT}^{\omega}_{(\mathrm{c})}$ ?
- Let us first establish an upper bound.
- A sequent $\Pi \vdash A$ is derivable from a set of hypotheses $\mathscr{H}$ if and only if $\Pi \vdash A$ belongs to **any** set of sequents including $\mathscr{H}$, including all axioms and closed under all rules of $\mathbf{ACT}^{\omega}_{(\mathrm{c})}$.
- This condition has one quantifier over **sets** of natural numbers (**second-order** arithmetical quantifier), followed by a property expressible in **first-order arithmetic.**
- Such conditions from a complexity class called $\Pi^1_1$ .

# Reasoning from Hypothesis

- What about derivability from hypotheses in $\mathbf{ACT}^{\omega}_{(\mathrm{c})}$ ?
- Let us first establish an upper bound.
- A sequent $\Pi \vdash A$ is derivable from a set of hypotheses $\mathscr{H}$ if and only if $\Pi \vdash A$ belongs to **any** set of sequents including $\mathscr{H}$, including all axioms and closed under all rules of $\mathbf{ACT}^{\omega}_{(\mathrm{c})}$.
- This condition has one quantifier over **sets** of natural numbers (**second-order** arithmetical quantifier), followed by a property expressible in **first-order arithmetic.**
- Such conditions from a complexity class called $\Pi^1_1$ .
- In fact, derivability from hypotheses in $\mathbf{ACT}^{\omega}_{(\mathrm{c})}$ is **$\Pi^1_1$-complete** [Kozen 2002; K. 2024].

# Reasoning from Hypothesis

- What about derivability from hypotheses in $\mathbf{ACT}^{\omega}_{(\mathrm{c})}$ ?
- Let us first establish an upper bound.
- A sequent $\Pi \vdash A$ is derivable from a set of hypotheses $\mathscr{H}$ if and only if $\Pi \vdash A$ belongs to **any** set of sequents including $\mathscr{H}$, including all axioms and closed under all rules of $\mathbf{ACT}^{\omega}_{(\mathrm{c})}$.
- This condition has one quantifier over **sets** of natural numbers (**second-order** arithmetical quantifier), followed by a property expressible in **first-order arithmetic.**
- Such conditions from a complexity class called $\Pi^1_1$ .
- In fact, derivability from hypotheses in $\mathbf{ACT}^{\omega}_{(\mathrm{c})}$ is **$\Pi^1_1$-complete** [Kozen 2002; K. 2024].
- This is proved by encoding well-foundedness of computable relations on $\omega$.

# Reasoning from $*$-Free Hypotheses

- Now let us restrict ourself to the case where hypotheses do not contain Kleene star.

# Reasoning from $*$-Free Hypotheses

- Now let us restrict ourself to the case where hypotheses do not contain Kleene star.
- The encoding of well-foundedness is no longer possible, but we still can encode both the halting and the non-halting problems.

# Reasoning from $*$-Free Hypotheses

- Now let us restrict ourself to the case where hypotheses do not contain Kleene star.
- The encoding of well-foundedness is no longer possible, but we still can encode both the halting and the non-halting problems.
- Moreover, the **totality** problem for a universal computational model (e.g., Turing machines) can also be encoded.

# Reasoning from $*$-Free Hypotheses

- Now let us restrict ourself to the case where hypotheses do not contain Kleene star.
- The encoding of well-foundedness is no longer possible, but we still can encode both the halting and the non-halting problems.
- Moreover, the **totality** problem for a universal computational model (e.g., Turing machines) can also be encoded.
- The totality problem is 'the machine halts on **any** input'; this problem is $\Pi^0_2$-complete.

# Reasoning from $*$-Free Hypotheses

- Now let us restrict ourself to the case where hypotheses do not contain Kleene star.
- The encoding of well-foundedness is no longer possible, but we still can encode both the halting and the non-halting problems.
- Moreover, the **totality** problem for a universal computational model (e.g., Turing machines) can also be encoded.
- The totality problem is 'the machine halts on **any** input'; this problem is $\Pi^0_2$-complete.
- In $\mathscr{H}$, we encode the machine, such that $E \vdash F$ is derivable from $\mathscr{H}$ if and only if the machine reaches the final state $F$ from the initial state $E$.

# Reasoning from $*$-Free Hypotheses

- Now let us restrict ourself to the case where hypotheses do not contain Kleene star.
- The encoding of well-foundedness is no longer possible, but we still can encode both the halting and the non-halting problems.
- Moreover, the **totality** problem for a universal computational model (e.g., Turing machines) can also be encoded.
- The totality problem is 'the machine halts on **any** input'; this problem is $\Pi^0_2$-complete.
- In $\mathscr{H}$, we encode the machine, such that $E \vdash F$ is derivable from $\mathscr{H}$ if and only if the machine reaches the final state $F$ from the initial state $E$.
- Now totality is encoded as $E, \mathtt{a}^* \vdash F$ (derivable from $\mathscr{H}$).

# Reasoning from $*$-Free Hypotheses

- The 'key-and-lock' technique allows encoding the whole **arithmetical hierarchy,** i.e., algorithmic properties of the form:

  $P(x) = \forall y_1\, \exists y_2\, \ldots\, \forall y_{2k+1}\, \big(\text{the machine halts on } \mathtt{a}_1^{y_1}\mathtt{a}_2^{y_2}\ldots\mathtt{a}_{2k+1}^{y_{2k+1}}\mathtt{b}^x\big).$

# Reasoning from $*$-Free Hypotheses

- The 'key-and-lock' technique allows encoding the whole **arithmetical hierarchy,** i.e., algorithmic properties of the form:

  $P(x) = \forall y_1\, \exists y_2\, \ldots\, \forall y_{2k+1}\, \big(\text{the machine halts on } \mathtt{a}_1^{y_1}\mathtt{a}_2^{y_2}\ldots\mathtt{a}_{2k+1}^{y_{2k+1}}\mathtt{b}^x\big).$

- For each $\forall y_{2i+1}$ quantifier, we have $q_{2i} \setminus (\mathtt{a}_{2i+1}^* \otimes q_{2i+1})$ in the sequent.

# Reasoning from $*$-Free Hypotheses

- The 'key-and-lock' technique allows encoding the whole **arithmetical hierarchy,** i.e., algorithmic properties of the form:

  $P(x) = \forall y_1\, \exists y_2\, \ldots\, \forall y_{2k+1}\, \big(\text{the machine halts on } \mathtt{a}_1^{y_1}\mathtt{a}_2^{y_2}\ldots\mathtt{a}_{2k+1}^{y_{2k+1}}\mathtt{b}^x\big).$

- For each $\forall y_{2i+1}$ quantifier, we have $q_{2i} \setminus (\mathtt{a}_{2i+1}^* \otimes q_{2i+1})$ in the sequent.
- For each $\exists y_{2i}$ quantifier, we have hypotheses $\vdash q_{2i-1} \setminus q_{2i}$ and $\vdash q_{2i} \setminus (\mathtt{a}_{2i} \otimes q_{2i})$.

# Reasoning from $*$-Free Hypotheses

- The 'key-and-lock' technique allows encoding the whole **arithmetical hierarchy,** i.e., algorithmic properties of the form:

  $P(x) = \forall y_1\, \exists y_2\, \ldots\, \forall y_{2k+1}\, \big(\text{the machine halts on } \mathtt{a}_1^{y_1}\mathtt{a}_2^{y_2}\ldots\mathtt{a}_{2k+1}^{y_{2k+1}}\mathtt{b}^x\big).$

- For each $\forall y_{2i+1}$ quantifier, we have $q_{2i} \setminus (\mathtt{a}_{2i+1}^* \otimes q_{2i+1})$ in the sequent.
- For each $\exists y_{2i}$ quantifier, we have hypotheses $\vdash q_{2i-1} \setminus q_{2i}$ and $\vdash q_{2i} \setminus (\mathtt{a}_{2i} \otimes q_{2i})$.
- In the original sequent, we have the first key $q_0$, and the input of the machine is protected by the last lock: $q_{2k+1} \setminus E$.

# Reasoning from $*$-Free Hypotheses

- The 'key-and-lock' technique allows encoding the whole **arithmetical hierarchy,** i.e., algorithmic properties of the form:

  $P(x) = \forall y_1\, \exists y_2\, \ldots\, \forall y_{2k+1}\, \big(\text{the machine halts on } \mathtt{a}_1^{y_1}\mathtt{a}_2^{y_2}\ldots\mathtt{a}_{2k+1}^{y_{2k+1}}\mathtt{b}^x\big).$

- For each $\forall y_{2i+1}$ quantifier, we have $q_{2i} \setminus (\mathtt{a}_{2i+1}^* \otimes q_{2i+1})$ in the sequent.
- For each $\exists y_{2i}$ quantifier, we have hypotheses $\vdash q_{2i-1} \setminus q_{2i}$ and $\vdash q_{2i} \setminus (\mathtt{a}_{2i} \otimes q_{2i})$.
- In the original sequent, we have the first key $q_0$, and the input of the machine is protected by the last lock: $q_{2k+1} \setminus E$.
- In fact, even this is not all! Reasoning from $*$-free hypothesis in $\mathbf{ACT}^\omega$ is $\Sigma^0_{\omega^\omega}$-complete [K., Pshenitsyn, Speranski 2025]. ($\omega^\omega$ iteration of Turing jump, or infinitary quantifier alternation.)

# Systems with Contraction

- As noticed yesterday, adding the **contraction** rule raises complexity: $\mathbf{FL}_{\mathrm{c}}$ is undecidable [Chvalovský, Horčík 2016] and $\mathbf{FL}_{\mathrm{ec}}$ is decidable [Kripke 1959], but not primitive recursive.

$$\frac{\Gamma, \Pi, \Pi, \Delta \vdash C}{\Gamma, \Pi, \Delta \vdash C}\ \mathrm{MC}$$

# Systems with Contraction

- As noticed yesterday, adding the **contraction** rule raises complexity: $\mathbf{FL}_{\mathrm{c}}$ is undecidable [Chvalovský, Horčík 2016] and $\mathbf{FL}_{\mathrm{ec}}$ is decidable [Kripke 1959], but not primitive recursive.

$$\frac{\Gamma, \Pi, \Pi, \Delta \vdash C}{\Gamma, \Pi, \Delta \vdash C}\ \mathrm{MC}$$

- The argument by Chvalovský and Horčík follows the general idea of encoding word rewriting systems, but faces two challenges.

# Systems with Contraction

- As noticed yesterday, adding the **contraction** rule raises complexity: $\mathbf{FL}_{\mathrm{c}}$ is undecidable [Chvalovský, Horčík 2016] and $\mathbf{FL}_{\mathrm{ec}}$ is decidable [Kripke 1959], but not primitive recursive.

$$\frac{\Gamma, \Pi, \Pi, \Delta \vdash C}{\Gamma, \Pi, \Delta \vdash C}\ \mathrm{MC}$$

- The argument by Chvalovský and Horčík follows the general idea of encoding word rewriting systems, but faces two challenges.
  - Contraction is local, so there is no way of moving the code of a production rules. Solution: conditional rewriting systems, i.e., context-free grammars with regular constraints on contexts.

# Systems with Contraction

- As noticed yesterday, adding the **contraction** rule raises complexity: $\mathbf{FL}_{\mathrm{c}}$ is undecidable [Chvalovský, Horčík 2016] and $\mathbf{FL}_{\mathrm{ec}}$ is decidable [Kripke 1959], but not primitive recursive.

$$\frac{\Gamma, \Pi, \Pi, \Delta \vdash C}{\Gamma, \Pi, \Delta \vdash C}\ \mathrm{MC}$$

- The argument by Chvalovský and Horčík follows the general idea of encoding word rewriting systems, but faces two challenges.
  - Contraction is local, so there is no way of moving the code of a production rules. Solution: conditional rewriting systems, i.e., context-free grammars with regular constraints on contexts.
  - Contraction is spurious: it can be applied to any word. This is resolved by considering only square-free words.

# Systems with Contraction

- As noticed yesterday, adding the **contraction** rule raises complexity: $\mathbf{FL}_{\mathrm{c}}$ is undecidable [Chvalovský, Horčík 2016] and $\mathbf{FL}_{\mathrm{ec}}$ is decidable [Kripke 1959], but not primitive recursive.

$$\frac{\Gamma, \Pi, \Pi, \Delta \vdash C}{\Gamma, \Pi, \Delta \vdash C}\ \mathrm{MC}$$

- The argument by Chvalovský and Horčík follows the general idea of encoding word rewriting systems, but faces two challenges.
  - Contraction is local, so there is no way of moving the code of a production rules. Solution: conditional rewriting systems, i.e., context-free grammars with regular constraints on contexts.
  - Contraction is spurious: it can be applied to any word. This is resolved by considering only square-free words.
- This argument is further extended to $\mathbf{ACT}_{\mathrm{c}}^{\omega}$, yielding its $\Pi_1^1$-completeness [K., Pshenitsyn 2026].

# Systems with Contraction

- As noticed yesterday, adding the **contraction** rule raises complexity: $\mathbf{FL}_{\mathrm{c}}$ is undecidable [Chvalovský, Horčík 2016] and $\mathbf{FL}_{\mathrm{ec}}$ is decidable [Kripke 1959], but not primitive recursive.

$$\frac{\Gamma, \Pi, \Pi, \Delta \vdash C}{\Gamma, \Pi, \Delta \vdash C}\ \mathrm{MC}$$

- The argument by Chvalovský and Horčík follows the general idea of encoding word rewriting systems, but faces two challenges.
  - Contraction is local, so there is no way of moving the code of a production rules. Solution: conditional rewriting systems, i.e., context-free grammars with regular constraints on contexts.
  - Contraction is spurious: it can be applied to any word. This is resolved by considering only square-free words.
- This argument is further extended to $\mathbf{ACT}_{\mathrm{c}}^{\omega}$, yielding its $\Pi_1^1$-completeness [K., Pshenitsyn 2026].
- In contrast, $\mathbf{ACT}_{\mathrm{ec}}^{\omega}$ is $\Pi_1^0$-complete [Greati, Ramanayake 2026].

- W. Buszkowski, E. Palka. Infinitary action logic: complexity, models and grammars. *Studia Logica,* **89**:1 (2008), 1–18.
- G. Morrill. Parsing/theorem-proving for logical grammar *CatLog3. J. Logic Lang. Inform.,* **28**:2 (2019), 183–216.
- C. Chvalovský, R. Horčík. Full Lambek calculus with contraction is undecidable. *J. Symb. Logic,* **81**:2 (2016), 525–540.
- A. Das, D. Pous. Non-wellfounded proof theory for (Kleene+action) (algebras+lattices). In: 27th EACSL Annual Conference on Computer Science Logic (CSL 2018), *Leibniz Internat. Proc. Inform.,* vol. **119** (2018), 19:1–19:18.
- S. L. Kuznetsov. Algorithmic complexity of theories with Kleene iteration. *Russian Math. Surveys,* **81**:1 (2026), 125–188.
- S. L. Kuznetsov, T. Pshenitsyn, S. O. Speranski. Reasoning from hypotheses in $*$-continuous action lattices. *J. Symb. Logic,* 2025, FirstView (published online).

- W. Buszkowski, E. Palka. Infinitary action logic: complexity, models and grammars. *Studia Logica,* **89**:1 (2008), 1–18.
- G. Morrill. Parsing/theorem-proving for logical grammar *CatLog3. J. Logic Lang. Inform.,* **28**:2 (2019), 183–216.
- C. Chvalovský, R. Horčík. Full Lambek calculus with contraction is undecidable. *J. Symb. Logic,* **81**:2 (2016), 525–540.
- A. Das, D. Pous. Non-wellfounded proof theory for (Kleene+action) (algebras+lattices). In: 27th EACSL Annual Conference on Computer Science Logic (CSL 2018), *Leibniz Internat. Proc. Inform.,* vol. **119** (2018), 19:1–19:18.
- S. L. Kuznetsov. Algorithmic complexity of theories with Kleene iteration. *Russian Math. Surveys,* **81**:1 (2026), 125–188.
- S. L. Kuznetsov, T. Pshenitsyn, S. O. Speranski. Reasoning from hypotheses in $*$-continuous action lattices. *J. Symb. Logic,* 2025, FirstView (published online).



`https://www.mi-ras.ru/~sk/` `sk@mi-ras.ru`

- W. Buszkowski, E. Palka. Infinitary action logic: complexity, models and grammars. *Studia Logica,* **89**:1 (2008), 1–18.
- G. Morrill. Parsing/theorem-proving for logical grammar *CatLog3. J. Logic Lang. Inform.,* **28**:2 (2019), 183–216.
- C. Chvalovský, R. Horčík. Full Lambek calculus with contraction is undecidable. *J. Symb. Logic,* **81**:2 (2016), 525–540.
- A. Das, D. Pous. Non-wellfounded proof theory for (Kleene+action) (algebras+lattices). In: 27th EACSL Annual Conference on Computer Science Logic (CSL 2018), *Leibniz Internat. Proc. Inform.,* vol. **119** (2018), 19:1–19:18.
- S. L. Kuznetsov. Algorithmic complexity of theories with Kleene iteration. *Russian Math. Surveys,* **81**:1 (2026), 125–188.
- S. L. Kuznetsov, T. Pshenitsyn, S. O. Speranski. Reasoning from hypotheses in $*$-continuous action lattices. *J. Symb. Logic,* 2025, FirstView (published online).



`https://www.mi-ras.ru/~sk/` `sk@mi-ras.ru`


# გმადლობთ