Complexity of Reasoning in Kleene and Action Algebras

Stepan Kuznetsov

ESSLLI 2022 · Galway, Ireland · Gallimh, Éire

Lecture 4

Divide and Conquer

- Now let us add **division** operators to Kleene algebras.

Divide and Conquer

- Now let us add **division** operators to Kleene algebras.
- $A \preceq C \,/\, B \iff A \cdot B \preceq C \iff B \preceq A \setminus C$

Divide and Conquer

- Now let us add **division** operators to Kleene algebras.
- $A \preceq C \,/\, B \iff A \cdot B \preceq C \iff B \preceq A \setminus C$
- Divisions have a natural language interpretation:

$$A \Big\backslash B = \{\beta \mid (\forall \alpha \in A)\, \alpha\beta \in B\}$$
$$B \Big/ A = \{\beta \mid (\forall \alpha \in A)\, \beta\alpha \in B\}$$

Divide and Conquer

- Now let us add **division** operators to Kleene algebras.
- $A \preceq C \,/\, B \iff A \cdot B \preceq C \iff B \preceq A \setminus C$
- Divisions have a natural language interpretation:

$$A \Big\backslash B = \{\beta \mid (\forall \alpha \in A)\, \alpha\beta \in B\}$$
$$B \Big/ A = \{\beta \mid (\forall \alpha \in A)\, \beta\alpha \in B\}$$

- Kleene algebras with divisions (residuated Kleene algebras) are called **action algebras.**

Divide and Conquer

- Now let us add **division** operators to Kleene algebras.
- $A \preceq C \,/\, B \iff A \cdot B \preceq C \iff B \preceq A \,\backslash\, C$
- Divisions have a natural language interpretation:

$$A \Big\backslash B = \{\beta \mid (\forall \alpha \in A)\, \alpha\beta \in B\}$$
$$B \Big/ A = \{\beta \mid (\forall \alpha \in A)\, \beta\alpha \in B\}$$

- Kleene algebras with divisions (residuated Kleene algebras) are called **action algebras.**
- Adding conjunction—intersection yields **action lattices.**

Divide and Conquer

- Now let us add **division** operators to Kleene algebras.
- $A \preceq C \,/\, B \iff A \cdot B \preceq C \iff B \preceq A \setminus C$
- Divisions have a natural language interpretation:

$$A \Big\backslash B = \{\beta \mid (\forall \alpha \in A)\, \alpha\beta \in B\}$$
$$B \Big/ A = \{\beta \mid (\forall \alpha \in A)\, \beta\alpha \in B\}$$

- Kleene algebras with divisions (residuated Kleene algebras) are called **action algebras.**
- Adding conjunction—intersection yields **action lattices.**
- The logic of all action lattices is action logic **ACT**.

Divide and Conquer

- Now let us add **division** operators to Kleene algebras.
- $A \preceq C \,/\, B \iff A \cdot B \preceq C \iff B \preceq A \setminus C$
- Divisions have a natural language interpretation:

$$A \Big\backslash B = \{\beta \mid (\forall \alpha \in A)\, \alpha\beta \in B\}$$
$$B \Big/ A = \{\beta \mid (\forall \alpha \in A)\, \beta\alpha \in B\}$$

- Kleene algebras with divisions (residuated Kleene algebras) are called **action algebras.**
- Adding conjunction—intersection yields **action lattices.**
- The logic of all action lattices is action logic **ACT**.
- The logic of *-continuous action lattices is infinitary action logic $\mathbf{ACT}_\omega$.

Action Lattices

- An **action lattice** is an algebraic structure $(\mathscr{A}; \preceq, \backslash, /, \vee, \wedge, 0, 1, {}^*)$, where:

Action Lattices

- An **action lattice** is an algebraic structure $(\mathscr{A}; \preceq, \backslash, /, \vee, \wedge, 0, 1, {}^*)$, where:
    1. $(\mathscr{A}; \preceq, \vee, \wedge)$ is a lattice, 0 is its least element;

Action Lattices

- An **action lattice** is an algebraic structure $(\mathscr{A}; \preceq, \backslash, /, \vee, \wedge, 0, 1, {}^*)$, where:
    1. $(\mathscr{A}; \preceq, \vee, \wedge)$ is a lattice, 0 is its least element;
    2. $(\mathscr{A}; \cdot, 1)$ is a monoid;

Action Lattices

- An **action lattice** is an algebraic structure $(\mathscr{A}; \preceq, \backslash, /, \vee, \wedge, 0, 1, {}^*)$, where:
  1. $(\mathscr{A}; \preceq, \vee, \wedge)$ is a lattice, 0 is its least element;
  2. $(\mathscr{A}; \cdot, 1)$ is a monoid;
  3. $\backslash$ and $/$ are residuals of the product w.r.t. $\preceq$:
     $A \preceq C \,/\, B \iff A \cdot B \preceq C \iff B \preceq A \,\backslash\, C$;

Action Lattices

- An **action lattice** is an algebraic structure $(\mathscr{A}; \preceq, \backslash, /, \vee, \wedge, 0, 1, {}^*)$, where:
    1. $(\mathscr{A}; \preceq, \vee, \wedge)$ is a lattice, 0 is its least element;
    2. $(\mathscr{A}; \cdot, 1)$ is a monoid;
    3. $\backslash$ and $/$ are residuals of the product w.r.t. $\preceq$:
       $A \preceq C \,/\, B \iff A \cdot B \preceq C \iff B \preceq A \,\backslash\, C$;
    4. $A^* = \min_{\preceq}\{b \mid 1 \preceq b \text{ and } a \cdot b \preceq b\}$.

Action Lattices

- An **action lattice** is an algebraic structure $(\mathscr{A}; \preceq, \backslash, /, \vee, \wedge, 0, 1, ^*)$, where:
    1. $(\mathscr{A}; \preceq, \vee, \wedge)$ is a lattice, 0 is its least element;
    2. $(\mathscr{A}; \cdot, 1)$ is a monoid;
    3. $\backslash$ and $/$ are residuals of the product w.r.t. $\preceq$:
       $A \preceq C / B \iff A \cdot B \preceq C \iff B \preceq A \backslash C$;
    4. $A^* = \min_{\preceq}\{b \mid 1 \preceq b \text{ and } a \cdot b \preceq b\}$.
- An action lattice is *-continuous, if $a^* = \sup_{\preceq}\{a^n \mid n \geq 0\}$.

Action Lattices

- In the presence of divisions, many things regularise:

Action Lattices

- In the presence of divisions, many things regularise:
    1. Product is monotone w.r.t. $\preceq$ (Lambek 1958).

Action Lattices

- In the presence of divisions, many things regularise:
  1. Product is monotone w.r.t. $\preceq$ (Lambek 1958).
  2. The left Kleene star is always the right one.

Action Lattices

- In the presence of divisions, many things regularise:
  1. Product is monotone w.r.t. $\preceq$ (Lambek 1958).
  2. The left Kleene star is always the right one.
  3. The zero element is the annihilator for product.

Action Lattices

- In the presence of divisions, many things regularise:
    1. Product is monotone w.r.t. $\preceq$ (Lambek 1958).
    2. The left Kleene star is always the right one.
    3. The zero element is the annihilator for product.
- The following **Pratt's normality theorem** holds: if there exists $\sup_{\preceq}\{a^n \mid n \geq 0\}$, then it should coincide with a^*.

Action Lattices

- In the presence of divisions, many things regularise:
  1. Product is monotone w.r.t. $\preceq$ (Lambek 1958).
  2. The left Kleene star is always the right one.
  3. The zero element is the annihilator for product.
- The following **Pratt's normality theorem** holds: if there exists $\sup_{\preceq}\{a^n \mid n \geq 0\}$, then it should coincide with a^*.
- This means that if the lattice is complete (allows infinite inf and sup), then it is *-continuous.

Action Lattices

- In the presence of divisions, many things regularise:
    1. Product is monotone w.r.t. $\preceq$ (Lambek 1958).
    2. The left Kleene star is always the right one.
    3. The zero element is the annihilator for product.
- The following **Pratt's normality theorem** holds: if there exists $\sup_{\preceq}\{a^n \mid n \geq 0\}$, then it should coincide with a^*.
- This means that if the lattice is complete (allows infinite inf and sup), then it is *-continuous.
- In particular, this holds for finite action lattices.

History and Motivation

- The first (probably) appearance of divisions (residuals) connected to a partial order is due to Krull (1924), who introduced them in algebra, for ideals in rings.

History and Motivation

- The first (probably) appearance of divisions (residuals) connected to a partial order is due to Krull (1924), who introduced them in algebra, for ideals in rings.
- **Residuated lattices** appear in the work of Ward & Dilworth (1939).

History and Motivation

- The first (probably) appearance of divisions (residuals) connected to a partial order is due to Krull (1924), who introduced them in algebra, for ideals in rings.
- **Residuated lattices** appear in the work of Ward & Dilworth (1939).
- Later on, Lambek (1958) introduced the Lambek calculus for defining natural language syntax.

History and Motivation

- The first (probably) appearance of divisions (residuals) connected to a partial order is due to Krull (1924), who introduced them in algebra, for ideals in rings.
- **Residuated lattices** appear in the work of Ward & Dilworth (1939).
- Later on, Lambek (1958) introduced the Lambek calculus for defining natural language syntax.
- The Lambek calculus is a basic substructural logic; on the connection of substructural logics and residuated lattices see Galatos et al. (2007).

History and Motivation

- Action algebras were introduced by Pratt (1991); Kozen (1994) added $\wedge$ and introduced action lattices.

History and Motivation

- Action algebras were introduced by Pratt (1991); Kozen (1994) added $\wedge$ and introduced action lattices.
- The motivation is in better properties of this class of algebras if compared with Kleene algebras.

History and Motivation

- Action algebras were introduced by Pratt (1991); Kozen (1994) added $\wedge$ and introduced action lattices.
- The motivation is in better properties of this class of algebras if compared with Kleene algebras.
- Namely, Kleene algebras do not form a finitely based variety (Redko 1964, Conway 1971), i.e., they cannot be axiomatised by a finite set of universally valid equations.

History and Motivation

- Action algebras were introduced by Pratt (1991); Kozen (1994) added $\wedge$ and introduced action lattices.
- The motivation is in better properties of this class of algebras if compared with Kleene algebras.
- Namely, Kleene algebras do not form a finitely based variety (Redko 1964, Conway 1971), i.e., they cannot be axiomatised by a finite set of universally valid equations.
- In contrast, action algebras do. Namely, as shown by Pratt, the condition for Kleene star can be replaced by **"pure induction"**

$$(A \setminus A)^* = A \setminus A,$$

and monotonicity: $A^* \preceq (A + B)^*$.

Standard Examples of Action Lattices

Standard Examples of Action Lattices

- $\mathscr{P}(\Sigma^*)$, the algebra of formal languages:
  - the lattice structure is set-theoretic;
  - $\cdot$ is pairwise concatenation, $1 = \{\varepsilon\}$;
  - $x \mathbin{/} y = \{u \in \Sigma^* \mid (\forall v \in y)\, uv \in x\}$,
    $y \setminus x = \{u \in \Sigma^* \mid (\forall v \in y)\, vu \in x\}$;
  - $x^* = \{u_1 \dots u_n \mid n \geq 0, u_i \in x\}$.

Standard Examples of Action Lattices

- $\mathscr{P}(\Sigma^*)$, the algebra of formal languages:
  - the lattice structure is set-theoretic;
  - $\cdot$ is pairwise concatenation, $1 = \{\varepsilon\}$;
  - $x \,/\, y = \{u \in \Sigma^* \mid (\forall v \in y)\, uv \in x\}$,
    $y \setminus x = \{u \in \Sigma^* \mid (\forall v \in y)\, vu \in x\}$;
  - $x^* = \{u_1 \ldots u_n \mid n \geq 0, u_i \in x\}$.
- $\mathscr{P}(W \times W)$, the algebra of relations:
  - the lattice structure is set-theoretic;
  - $\cdot$ is composition of relations, 1 is the diagonal;
  - $x \,/\, y = \{\langle a, b\rangle \mid (\forall \langle b, c\rangle \in y)\, \langle a, c\rangle \in x\}$,
    $y \setminus x = \{\langle b, c\rangle \mid (\forall \langle a, b\rangle \in y)\, \langle a, c\rangle \in x\}$;
  - x^* is the reflexive-transitive closure of x.

Standard Examples of Action Lattices

- $\mathscr{P}(\Sigma^*)$, the algebra of formal languages:
    - the lattice structure is set-theoretic;
    - $\cdot$ is pairwise concatenation, $1 = \{\varepsilon\}$;
    - $x \,/\, y = \{u \in \Sigma^* \mid (\forall v \in y)\, uv \in x\}$,
      $y \setminus x = \{u \in \Sigma^* \mid (\forall v \in y)\, vu \in x\}$;
    - $x^* = \{u_1 \ldots u_n \mid n \geq 0, u_i \in x\}$.
- $\mathscr{P}(W \times W)$, the algebra of relations:
    - the lattice structure is set-theoretic;
    - $\cdot$ is composition of relations, 1 is the diagonal;
    - $x \,/\, y = \{\langle a, b\rangle \mid (\forall \langle b, c\rangle \in y)\, \langle a, c\rangle \in x\}$,
      $y \setminus x = \{\langle b, c\rangle \mid (\forall \langle a, b\rangle \in y)\, \langle a, c\rangle \in x\}$;
    - x^* is the reflexive-transitive closure of x.

These action lattices are *distributive* (as lattices) and **-continuous.*

Action Lattices

- However, non-*-continuous action lattices exist.

Action Lattices

- However, non-*-continuous action lattices exist.
- This is an example, modifying Kozen's one (1990) by adding divisions and avoiding suprema:

$$\begin{array}{cccccc} \bullet & \longmapsto & \longleftrightarrow & \longleftrightarrow & \cdots & \bullet \\ \bot & \mathbb{N} & \mathbb{Z} & \mathbb{Z} & & \top \end{array}$$

Action Lattices

- However, non-*-continuous action lattices exist.
- This is an example, modifying Kozen's one (1990) by adding divisions and avoiding suprema:

$$\begin{array}{cccccc} \bullet & \longmapsto & \longleftrightarrow & \longleftrightarrow & \cdots & \bullet \\ \bot & \mathbb{N} & \mathbb{Z} & \mathbb{Z} & & \top \end{array}$$

  - Multiplication: componentwise addition in the middle; $\bot \cdot x = x \cdot \bot = \bot$; $\top \cdot y = y \cdot \top = \top$ for $y \neq \bot$.

Action Lattices

- However, non-*-continuous action lattices exist.
- This is an example, modifying Kozen's one (1990) by adding divisions and avoiding suprema:

$$\begin{array}{cccccc} \bullet & \longmapsto & \longleftrightarrow & \longleftrightarrow & \cdots & \bullet \\ \bot & \mathbb{N} & \mathbb{Z} & \mathbb{Z} & & \top \end{array}$$

  - Multiplication: componentwise addition in the middle; $\bot \cdot x = x \cdot \bot = \bot$; $\top \cdot y = y \cdot \top = \top$ for $y \neq \bot$.
  - Unit: $1 = (0, 0)$.

- However, non-*-continuous action lattices exist.
- This is an example, modifying Kozen's one (1990) by adding divisions and avoiding suprema:

$$\begin{array}{cccccc} \bullet & \longmapsto & \longleftrightarrow & \longleftrightarrow & \cdots & \bullet \\ \bot & \mathbb{N} & \mathbb{Z} & \mathbb{Z} & & \top \end{array}$$

  - Multiplication: componentwise addition in the middle; $\bot \cdot x = x \cdot \bot = \bot$; $\top \cdot y = y \cdot \top = \top$ for $y \neq \bot$.
  - Unit: $1 = (0, 0)$.
  - One can show that this monoid is residuated.

Action Lattices

- However, non-*-continuous action lattices exist.
- This is an example, modifying Kozen's one (1990) by adding divisions and avoiding suprema:

$$\begin{array}{cccccc} \bullet & \longmapsto & \longleftrightarrow & \longleftrightarrow & \cdots & \bullet \\ \bot & \mathbb{N} & \mathbb{Z} & \mathbb{Z} & & \top \end{array}$$

  - Multiplication: componentwise addition in the middle; $\bot \cdot x = x \cdot \bot = \bot$; $\top \cdot y = y \cdot \top = \top$ for $y \neq \bot$.
  - Unit: $1 = (0,0)$.
  - One can show that this monoid is residuated.
  - $x^* = \top$ for $x \succ (0,0)$; $\bot^* = (0,0)^* = (0,0)$.

Action Lattices

- However, non-*-continuous action lattices exist.
- This is an example, modifying Kozen's one (1990) by adding divisions and avoiding suprema:

$$
\begin{array}{cccccc}
\bullet & \longmapsto & \longleftrightarrow & \longleftrightarrow & \cdots & \bullet \\
\bot & \mathbb{N} & \mathbb{Z} & \mathbb{Z} & & \top
\end{array}
$$

  - Multiplication: componentwise addition in the middle; $\bot \cdot x = x \cdot \bot = \bot$; $\top \cdot y = y \cdot \top = \top$ for $y \neq \bot$.
  - Unit: $1 = (0, 0)$.
  - One can show that this monoid is residuated.
  - $x^* = \top$ for $x \succ (0, 0)$; $\bot^* = (0, 0)^* = (0, 0)$.
  - $\sup\{(0, 1)^n \mid n \geq 0\} = \sup\{(0, n) \mid n \geq 0\}$ does not exist.

Action Lattices

- However, non-*-continuous action lattices exist.
- This is an example, modifying Kozen's one (1990) by adding divisions and avoiding suprema:

$$\begin{array}{cccccc} \bullet & \longmapsto & \longleftrightarrow & \longleftrightarrow & \cdots & \bullet \\ \bot & \mathbb{N} & \mathbb{Z} & \mathbb{Z} & & \top \end{array}$$

  - Multiplication: componentwise addition in the middle; $\bot \cdot x = x \cdot \bot = \bot$; $\top \cdot y = y \cdot \top = \top$ for $y \neq \bot$.
  - Unit: $1 = (0, 0)$.
  - One can show that this monoid is residuated.
  - $x^* = \top$ for $x \succ (0, 0)$; $\bot^* = (0, 0)^* = (0, 0)$.
  - $\sup\{(0, 1)^n \mid n \geq 0\} = \sup\{(0, n) \mid n \geq 0\}$ does not exist.
- Extra properties: *commutativity* and *linearity* of $\preceq$.

Action Lattices

- Moreover, unlike the situation with Kleene algebras, for action lattices the **inequational theories** differ in the general case and in the *-continuous one.

Action Lattices

- Moreover, unlike the situation with Kleene algebras, for action lattices the **inequational theories** differ in the general case and in the *-continuous one.
- This will follow from complexity considerations below, but there is also an explicit example.

Action Lattices

- Moreover, unlike the situation with Kleene algebras, for action lattices the **inequational theories** differ in the general case and in the *-continuous one.
- This will follow from complexity considerations below, but there is also an explicit example.
- This example is based on the following **induction-in-the-middle** rule:

$$\frac{\to B \quad A \to B \quad A, B, A \to B}{A^* \to B} \; {*L_{\mathrm{mid}}}$$

Action Lattices

- Moreover, unlike the situation with Kleene algebras, for action lattices the **inequational theories** differ in the general case and in the *-continuous one.
- This will follow from complexity considerations below, but there is also an explicit example.
- This example is based on the following **induction-in-the-middle** rule:

$$\frac{\to B \quad A \to B \quad A, B, A \to B}{A^* \to B} \; {*}L_{\mathrm{mid}}$$

- This allows deriving the following sequent:

$$(p \wedge q \wedge (p \setminus q) \wedge (p \,/\, q))^+ \to p,$$

which can be falsified on a non-*-continuous action algebra.

Complexity of Action Logic

Theorem (Buszkowski & Palka 2007)

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

Theorem (K. 2019–20)

$\mathbf{ACT}$ *is* Σ^0_1*-complete.*

Complexity of Action Logic

Theorem (Buszkowski & Palka 2007)

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

Theorem (K. 2019–20)

$\mathbf{ACT}$ *is* Σ^0_1*-complete.*

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

Complexity of Action Logic

Theorem (Buszkowski & Palka 2007)

ACT$_\omega$ *is* Π^0_1*-complete.*

Theorem (K. 2019–20)

ACT *is* Σ^0_1*-complete.*

- For simplicity, in what follows we shift to the commutative situation, and consider **CommACT**$_\omega$ and **CommACT**.
- In the commutative case, we have only one division: $A \setminus B \equiv B \mathbin{/} A$.

Commutative Action Algebras/Lattices

A commutative action algebra is an action algebra satisfying $ab = ba$. Whereas action logic in general is neutral as to whether ab combines a and b sequentially or concurrently, commutative action logic in effect commits to concurrency.

Pratt (1991)

Multiplicative-additive Lambek Calculus

- The core of **ACT** and $\mathbf{ACT}_\omega$ is **MALC**, the inequational theory of residuated lattices.

Multiplicative-additive Lambek Calculus

- The core of **ACT** and $\mathbf{ACT}_\omega$ is **MALC**, the inequational theory of residuated lattices.
- We consider its commutative modification **CommMALC**, that is, left-hand sides are multisets.

Multiplicative-additive Lambek Calculus

- The core of **ACT** and **ACT**$_\omega$ is **MALC**, the inequational theory of residuated lattices.
- We consider its commutative modification **CommMALC**, that is, left-hand sides are multisets.
- Axioms and rules: $\dfrac{}{A \to A}\ Id$ $\qquad \dfrac{}{\Gamma, 0 \to B}\ 0L$

$$\frac{\Gamma \to B}{\Gamma, 1 \to B}\ 1L \qquad \frac{}{\to 1}\ 1R \qquad \frac{\Gamma, A, B \to C}{\Gamma, A \cdot B \to C}\ {\cdot}L \qquad \frac{\Gamma \to A \quad \Delta \to B}{\Gamma, \Delta \to A \cdot B}\ {\cdot}R$$

$$\frac{\Gamma, A \to C \quad \Gamma, B \to C}{\Gamma, A \vee B \to C}\ \vee L \qquad \frac{\Pi \to A}{\Pi \to A \vee B}\ \vee R_1 \qquad \frac{\Pi \to B}{\Pi \to A \vee B}\ \vee R_2$$

$$\frac{\Gamma, A \to C}{\Gamma, A \wedge B \to C}\ \wedge L_1 \qquad \frac{\Gamma, B \to C}{\Gamma, A \wedge B \to C}\ \wedge L_2 \qquad \frac{\Pi \to A \quad \Pi \to B}{\Pi \to A \wedge B}\ \wedge R$$

$$\frac{\Pi \to A \quad \Gamma, B \to C}{\Gamma, \Pi, A \setminus B \to C}\ \setminus L \qquad \frac{A, \Pi \to B}{\Pi \to A \setminus B}\ / L$$

Rules for Kleene Star

- In $\mathbf{CommACT}_\omega$:

$$\frac{\left(\Gamma, A^n \to C\right)_{n=0}^{\infty}}{\Gamma, A^* \to C} \ {*L_\omega} \qquad \frac{\Gamma_1 \to A \quad \dots \quad \Gamma_n \to A}{\Gamma_1, \dots, \Gamma_n \to A^*} \ {*R_n},\ n \geq 0$$

Rules for Kleene Star

- In $\mathbf{CommACT}_\omega$:

$$\frac{\left(\Gamma, A^n \to C\right)_{n=0}^{\infty}}{\Gamma, A^* \to C} *L_\omega \qquad \frac{\Gamma_1 \to A \quad \dots \quad \Gamma_n \to A}{\Gamma_1, \dots, \Gamma_n \to A^*} *R_n,\ n \geq 0$$

- In $\mathbf{CommACT}$:

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

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

Minsky Machines

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

Minsky Machines

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

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

Minsky Machines

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

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

- We consider only *deterministic* Minsky machines.

Minsky Machines

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

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

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

Minsky Machines

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

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

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

Minsky Machines

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

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

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

Encoding Minsky Instructions

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

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

Encoding Minsky Instructions

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

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

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

Encoding Minsky Instructions

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

$$
\begin{aligned}
A_{\text{INC}(p,r,q)} &= p \setminus (q \cdot r) \\
A_{\text{JZDEC}(p,r,q_0,q_1)} &= ((p \cdot r) \setminus q_1) \wedge (p \setminus (q_0 \vee z_r)).
\end{aligned}
$$

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

Encoding Minsky Instructions

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

$$A_{\mathrm{INC}(p,r,q)} = p \Big\backslash (q \cdot r)$$
$$A_{\mathrm{JZDEC}(p,r,q_0,q_1)} = ((p \cdot r) \Big\backslash q_1) \wedge (p \Big\backslash (q_0 \vee z_r)).$$

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

Encoding Minsky Instructions

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

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

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

Encoding Infinite Execution

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

Encoding Infinite Execution

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

Encoding Infinite Execution

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

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

Encoding Infinite Execution

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

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

↑

this formula encodes any valid configuration

Encoding Infinite Execution

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

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

↑

this formula encodes any valid configuration

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

Encoding Infinite Execution

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

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

$\uparrow$

this formula encodes any valid configuration

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

Lemma

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

Encoding Infinite Computation

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

Encoding Infinite Computation

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

Encoding Infinite Computation

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

Encoding Infinite Computation

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

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

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

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

- $a \neq 0$

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

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

- $a \neq 0$

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

- $a = 0$

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

Backwards Implication

- As usual, proving the backwards direction, from derivation to computation, is technically more involved.

Backwards Implication

- As usual, proving the backwards direction, from derivation to computation, is technically more involved.
- Again, we reduce to finite proofs of sequents $E^n, \mathrm{a}^a, \mathrm{b}^b, \mathrm{c}^c, q \to D$.

Backwards Implication

- As usual, proving the backwards direction, from derivation to computation, is technically more involved.
- Again, we reduce to finite proofs of sequents $E^n, \mathrm{a}^a, \mathrm{b}^b, \mathrm{c}^c, q \to D$.
- In fact, it is possible that such a proof does not directly correspond to a counter machine computation: it could include "subprograms."

- For example, let the machine include the following instructions: $\textsc{inc}(p, \mathrm{a}, q)$ and $\textsc{jzdec}(q, \mathrm{a}, p, p)$, and consider a 4-step execution starting from $(p, 0, 0, 0)$.
- This execution has the following "non-canonical" representation:

$$
\dfrac{\underline{\dfrac{
\dfrac{\underline{\dfrac{
p \to p \qquad
\dfrac{\underline{\dfrac{q, \mathrm{a} \to q \cdot \mathrm{a} \qquad p \to p}{(q \cdot \mathrm{a}) \setminus p, \mathrm{a}, q \to p}\ \setminus L}}{E, q, \mathrm{a} \to p}\ \wedge L
}{E, p \setminus (q \cdot \mathrm{a}), p \to p}\ \cdot L, \setminus L}}{E^2, p \to p}\ \wedge L
\qquad
\dfrac{\dfrac{\underline{\dfrac{q, \mathrm{a} \to q \cdot \mathrm{a} \qquad p \to D}{(q \cdot \mathrm{a}) \setminus p, q, \mathrm{a} \to D}\ \setminus L}}{E, q, \mathrm{a} \to D}\ \wedge L}{E, q \cdot \mathrm{a} \to D}\ \cdot L
}{E^3, p \setminus (q \cdot \mathrm{a}), p \to D}\ \setminus L}}{E^4, p \to D}\ \wedge L
$$

Backwards Implication

- Here we perform the INC step, and then start a “subroutine” which performs INC and JZDEC, returning to the same state. Finally, we perform JZDEC.

Backwards Implication

- Here we perform the INC step, and then start a "subroutine" which performs INC and JZDEC, returning to the same state. Finally, we perform JZDEC.
- The crucial point is that such a "subroutine" could not use the zero branch of JZDEC.

Backwards Implication

- Here we perform the INC step, and then start a "subroutine" which performs INC and JZDEC, returning to the same state. Finally, we perform JZDEC.
- The crucial point is that such a "subroutine" could not use the zero branch of JZDEC.
- This is due to the fact that D is available only on the main branch.

Backwards Implication

Let $\widetilde{E}_i$ denote any formula in the conjunction E or a conjunction of such formulae. The backwards implication is proved by joint induction of 6 statements:

1. Sequents of the form $\widetilde{E}_1, \dots, \widetilde{E}_k, \mathrm{a}^a, \mathrm{b}^b, \mathrm{c}^c \to t$, where $t \in Q \cup Z$, are never derivable, neither are sequents of the form $\widetilde{E}_1, \dots, \widetilde{E}_k, \mathrm{a}^a, \mathrm{b}^b, \mathrm{c}^c \to t \cdot r$, where $r \in R$.
2. Sequents of the form $\widetilde{E}_1, \dots, \widetilde{E}_k, z_r, \mathrm{a}^a, \mathrm{b}^b, \mathrm{c}^c \to t$, where $r \in R$ and $t \in Q \cup Z_{\bar{r}}$, are never derivable, neither are sequents of the form $\widetilde{E}_1, \dots, \widetilde{E}_k, z_r, \mathrm{a}^a, \mathrm{b}^b, \mathrm{c}^c \to t \cdot r'$, where $r, r' \in R$ and $t \in Q \cup Z_{\bar{r}}$.
3. If $\widetilde{E}_1, \dots, \widetilde{E}_k, z_{\mathrm{a}}, \mathrm{a}^a, \mathrm{b}^b, \mathrm{c}^c \to D$ is derivable, then $a = 0$. Similarly for b and c.

Backwards Implication

4. If $\widetilde{E}_1, \ldots, \widetilde{E}_k, q, \mathrm{a}^{a'}, \mathrm{b}^{b'}, \mathrm{c}^{c'} \to p$ is derivable, where $p, q \in Q$, then the machine can move from $\langle q, a' + a, b' + b, c' + c\rangle$ to $\langle p, a, b, c\rangle$ in k steps for any a, b, c.
5. If $\widetilde{E}_1, \ldots, \widetilde{E}_k, q, \mathrm{a}^{a'}, \mathrm{b}^{b'}, \mathrm{c}^{c'} \to p \cdot \mathrm{a}$, where $p, q \in Q$, is derivable, then the machine can move from $\langle q, a' + a, b' + b, c' + c\rangle$ to $\langle p, a + 1, b, c\rangle$ in k steps for any a, b, c. Similarly for b and c.
6. If $\widetilde{E}_1, \ldots, \widetilde{E}_k, p, \mathrm{a}^{a}, \mathrm{b}^{b}, \mathrm{c}^{c} \to D$ is derivable ($p \in Q$), then the machine can perform k steps, starting from $\langle p, a, b, c\rangle$.

- Thus, derivability in **CommACT** is capable of simulating non-halting for counter machines, which yields the commutative version of Buszkowski's theorem:

- Thus, derivability in **CommACT** is capable of simulating non-halting for counter machines, which yields the commutative version of Buszkowski's theorem:

Theorem

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

- Thus, derivability in **CommACT** is capable of simulating non-halting for counter machines, which yields the commutative version of Buszkowski's theorem:

Theorem

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

- For the Π^0_1 upper bound, there are several methods, one of which we shall discuss tomorrow.

Undecidability Results

- Thus, derivability in **CommACT** is capable of simulating non-halting for counter machines, which yields the commutative version of Buszkowski's theorem:

Theorem

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

- For the Π^0_1 upper bound, there are several methods, one of which we shall discuss tomorrow.
- Next, we aim to prove undecidability of **CommACT**.

Undecidability Results

- Thus, derivability in **CommACT** is capable of simulating non-halting for counter machines, which yields the commutative version of Buszkowski's theorem:

Theorem

CommACT$_\omega$ *is* Π^0_1*-hard.*

- For the Π^0_1 upper bound, there are several methods, one of which we shall discuss tomorrow.
- Next, we aim to prove undecidability of **CommACT**.
- This will be done by encoding circular computations by circular proofs.

Lemma

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

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

Translating Circular Proofs to CommACT

- Now we have to translate this circular argument into a proof in **CommACT**.

Translating Circular Proofs to CommACT

- Now we have to translate this circular argument into a proof in **CommACT**.
- Recall the rule for * there:

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

Translating Circular Proofs to CommACT

- Now we have to translate this circular argument into a proof in **CommACT**.
- Recall the rule for * there:

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

- First we establish $*L$, the decomposition rule:

$$\dfrac{A^* \to 1 \vee (A \cdot A^*) \qquad \dfrac{\dfrac{\Gamma \to C}{\Gamma, 1 \to C}\,1L \qquad \dfrac{\Gamma, A, A^* \to C}{\Gamma, A \cdot A^* \to C}\,\cdot L}{\Gamma, 1 \vee (A \cdot A^*) \to C}\,\vee L}{\Gamma, A^* \to C}\,Cut$$

Translating Circular Proofs to CommACT

- Now we have to translate this circular argument into a proof in **CommACT**.
- Recall the rule for * there:

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

- First we establish $*L$, the decomposition rule:

$$\dfrac{A^* \to 1 \vee (A \cdot A^*) \qquad \dfrac{\dfrac{\Gamma \to C}{\Gamma, 1 \to C}\,1L \qquad \dfrac{\Gamma, A, A^* \to C}{\Gamma, A \cdot A^* \to C}\,\cdot L}{\Gamma, 1 \vee (A \cdot A^*) \to C}\,\vee L}{\Gamma, A^* \to C}\,Cut$$

- This will be needed for modelling computation before the cycle.

Translating Circular Proofs to CommACT

- Next, we establish an extended version of induction:

$$\frac{\to B \quad A \to B \quad \dots \quad A^{k-1} \to B \quad A^k, B \to B}{A^* \to B}$$

Translating Circular Proofs to CommACT

- Next, we establish an extended version of induction:

$$\frac{\to B \quad A \to B \quad \dots \quad A^{k-1} \to B \quad A^k, B \to B}{A^* \to B}$$

- This is established by cutting with $A^* \equiv (1 \vee A \vee \dots \vee A^{k-1})(A^k)^*$.

Translating Circular Proofs to CommACT

- Next, we establish an extended version of induction:

$$\frac{\to B \quad A \to B \quad \ldots \quad A^{k-1} \to B \quad A^k, B \to B}{A^* \to B}$$

- This is established by cutting with $A^* \equiv (1 \vee A \vee \ldots \vee A^{k-1})(A^k)^*$.
- Now we have a circular proof of $E^*, p, \mathrm{a}^a, \mathrm{b}^b, \mathrm{c}^c \to D$ from itself, and let k be the number of $*L$ applications on the cycle.

Translating Circular Proofs to CommACT

- Next, we establish an extended version of induction:

$$\frac{\to B \quad A \to B \quad \dots \quad A^{k-1} \to B \quad A^k, B \to B}{A^* \to B}$$

- This is established by cutting with $A^* \equiv (1 \vee A \vee \dots \vee A^{k-1})(A^k)^*$.
- Now we have a circular proof of $E^*, p, \mathrm{a}^a, \mathrm{b}^b, \mathrm{c}^c \to D$ from itself, and let k be the number of $*L$ applications on the cycle.
- For simplicity, let $F = (p \cdot \mathrm{a}^a \cdot \mathrm{b}^b \cdot \mathrm{c}^c) \setminus D$. Then our sequent is equivalent to $E^* \to F$.

Translating Circular Proofs to CommACT

- Next, we establish an extended version of induction:

$$\frac{\to B \quad A \to B \quad \dots \quad A^{k-1} \to B \quad A^k, B \to B}{A^* \to B}$$

- This is established by cutting with $A^* \equiv (1 \vee A \vee \dots \vee A^{k-1})(A^k)^*$.
- Now we have a circular proof of $E^*, p, \mathrm{a}^a, \mathrm{b}^b, \mathrm{c}^c \to D$ from itself, and let k be the number of $*L$ applications on the cycle.
- For simplicity, let $F = (p \cdot \mathrm{a}^a \cdot \mathrm{b}^b \cdot \mathrm{c}^c) \setminus D$. Then our sequent is equivalent to $E^* \to F$.
- From the circular proof, we can easily extract $E^i \to F$ for $0 \leq i < k$ (by replacing E^* with E^i).

Translating Circular Proofs to CommACT

- Next, we replace E with F, making $E^* \to F$ on top an axiom.

Translating Circular Proofs to CommACT

- Next, we replace E with F, making $E^* \to F$ on top an axiom.
- When we descend to the root, we get $E^k, F \to F$.

Translating Circular Proofs to CommACT

- Next, we replace E with F, making $E^* \to F$ on top an axiom.
- When we descend to the root, we get $E^k, F \to F$.
- Appplying our extended induction rule, we get $E^* \to F$, q.e.d.

Translating Circular Proofs to CommACT

- Next, we replace E with F, making $E^* \to F$ on top an axiom.
- When we descend to the root, we get $E^k, F \to F$.
- Appplying our extended induction rule, we get $E^* \to F$, q.e.d.
- Due to lack of time, we omitted the zero-checks, which would also involve circular reasoning (or using "pure induction," $(z_\mathrm{a} \setminus z_\mathrm{a})^* = z_\mathrm{a} \setminus z_\mathrm{a}$).

Translating Circular Proofs to CommACT

- Next, we replace E with F, making $E^* \to F$ on top an axiom.
- When we descend to the root, we get $E^k, F \to F$.
- Appplying our extended induction rule, we get $E^* \to F$, q.e.d.
- Due to lack of time, we omitted the zero-checks, which would also involve circular reasoning (or using "pure induction," $(z_\mathrm{a} \setminus z_\mathrm{a})^* = z_\mathrm{a} \setminus z_\mathrm{a}$).
- In fact, circular proofs can always be rebuilt into inductive ones, but proving this in a general setting is much harder.

Inseparability

- Not any circular derivation, however, corresponds to a circular run of the machine.

Inseparability

- Not any circular derivation, however, corresponds to a circular run of the machine.
- For example, if the machine just increases one counter, $\textsc{inc}(q, \mathrm{a}, q)$, then $E^*, \mathrm{a}^a, \mathrm{b}^b, \mathrm{c}^c, q \to D$ is also derivable in **CommACT**.

- Not any circular derivation, however, corresponds to a circular run of the machine.
- For example, if the machine just increases one counter, $\textsc{inc}(q, \mathrm{a}, q)$, then $E^*, \mathrm{a}^a, \mathrm{b}^b, \mathrm{c}^c, q \to D$ is also derivable in **CommACT**.
- In this case $E = q \backslash (q \cdot \mathrm{a})$, and the circular derivation, for $a = b = c = 0$, is as follows:

$$
\dfrac{\dfrac{q_S \to \mathrm{a}^*\cdot q_S \quad \dfrac{\dfrac{q_S \to q_S \quad \dfrac{\dfrac{E^*, q_S \to \mathrm{a}^*\cdot q_S \quad \mathrm{a}\to\mathrm{a}}{E^*, q_S, \mathrm{a} \to \mathrm{a}\cdot(\mathrm{a}^*\cdot q_S)}\,{\cdot R} \quad \mathrm{a}\cdot(\mathrm{a}^*\cdot q_S)\to \mathrm{a}^*\cdot q_S}{E^*, q_S, \mathrm{a}\to \mathrm{a}^*\cdot q_S}\,\mathit{Cut}}{E^*, q_S \backslash (q_S\cdot \mathrm{a}), q_S \to \mathrm{a}^*\cdot q_S}\,{\cdot L}, \backslash L}{\overline{E^*, E, q_S \to \mathrm{a}^*\cdot q_S}}\,\wedge L}{E^*, q_S \to \mathrm{a}^* \cdot q_S}\,{*L} \quad \mathrm{a}^*\cdot q_S \to D}{E^*, q_S \to D}\,\mathit{Cut}
$$

Inseparability

- Therefore, we use an indirect technique for proving complexity, based on **effective inseparability.**

Inseparability

- Therefore, we use an indirect technique for proving complexity, based on **effective inseparability.**
- Let $\mathscr{C}$ be the set of machines and input data such that the machine works circularly, and let $\mathscr{H}$ be the one where the machine halts.

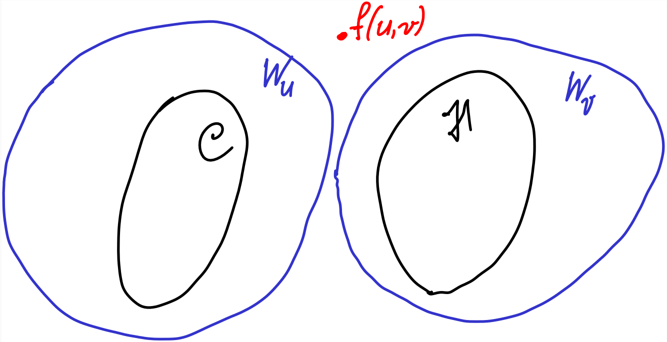
Inseparability

- Therefore, we use an indirect technique for proving complexity, based on **effective inseparability.**
- Let $\mathscr{C}$ be the set of machines and input data such that the machine works circularly, and let $\mathscr{H}$ be the one where the machine halts.
- There is no decidable set $\mathscr{K}$ which separates $\mathscr{C}$ from $\mathscr{H}$ (i.e., $\mathscr{C} \subseteq \mathscr{K}$ and $\mathscr{H} \cap \mathscr{K} = \emptyset$), therefore **CommACT** is undecidable; but we do not know its complexity yet.

Inseparability

- Therefore, we use an indirect technique for proving complexity, based on **effective inseparability.**
- Let $\mathscr{C}$ be the set of machines and input data such that the machine works circularly, and let $\mathscr{H}$ be the one where the machine halts.
- There is no decidable set $\mathscr{K}$ which separates $\mathscr{C}$ from $\mathscr{H}$ (i.e., $\mathscr{C} \subseteq \mathscr{K}$ and $\mathscr{H} \cap \mathscr{K} = \emptyset$), therefore **CommACT** is undecidable; but we do not know its complexity yet.
- **Folklore:** $\mathscr{C}$ and $\mathscr{H}$ are *effectively* inseparable.

Effective Inseparability



(Here W_u is the u-th r.e. set; f is computable.)

Myhill's Theorem

- (Corollary of) **Myhill's theorem:** if A separates $\mathscr{C}$ and $\mathscr{H}$ and A is r.e., then A is Σ^0_1-complete.

Myhill's Theorem

- (Corollary of) **Myhill's theorem:** if A separates $\mathscr{C}$ and $\mathscr{H}$ and A is r.e., then A is Σ^0_1-complete.
- Therefore, we obtain the necessary result:

Myhill's Theorem

- (Corollary of) **Myhill's theorem:** if A separates $\mathscr{C}$ and $\mathscr{H}$ and A is r.e., then A is Σ^0_1-complete.
- Therefore, we obtain the necessary result:

Theorem

CommACT *is* Σ^1_0*-complete.*

Myhill's Theorem

- (Corollary of) **Myhill's theorem:** if A separates $\mathscr{C}$ and $\mathscr{H}$ and A is r.e., then A is Σ_1^0-complete.
- Therefore, we obtain the necessary result:

Theorem

CommACT *is* Σ_0^1*-complete.*

- The reasoning in the non-commutative case is similar, however, the encoding of computations is more involved.