On the coinductive nature of centralizers

Charles Grellois

LIAFA & PPS — Université Paris 7

Séminaire Thésards — July 1, 2015

Commutation of words

Consider the word equation

$$x \cdot w \; = \; w \cdot x$$

Solution: a word x which is a prefix and a suffix of w.

Well-known fact: if

$$w \; = \; u^n$$

(u being minimal), the solutions are

$$\{\, u^m \mid m \in \mathbb{N} \,\}$$

Commutation of words

Example:

$$x \cdot abab \;=\; abab \cdot x$$

Solutions: $\{\, (ab)^m \mid m \in \mathbb{N} \,\}$.

Generalization to polynomials or formal series (variables do not commute).

What about commutation of languages ?

Centralizers

Consider the language commutation equation

$$X \cdot L \; = \; L \cdot X \tag{1}$$

X solution iff for all $(w, x) \; \in \; L \times X$,

$$w \cdot x \quad \in L \cdot X$$

can be factored as

$$x' \cdot w' \quad \in X \cdot L$$

with $x' \in X$ and $w' \in L$.

(and conversely)

Centralizers

Consider the language commutation equation

$$X \cdot L \; = \; L \cdot X \tag{1}$$

X solution iff for all $(w, x) \;\in\; L \times X$,

$$w \cdot x \quad \in L \cdot X$$

can be factored as

$$x' \cdot w' \quad \in X \cdot L$$

with $x' \in X$ and $w' \in L$.

(and conversely)

Centralizers: an example

Consider (from Choffrut-Karhumäki-Ollinger):

$$L \quad = \quad \{\, a,\, a^3,\, b,\, ba,\, ab,\, aba \,\}$$

then a solution of the commutation equation is

$$X \quad = \quad L \cup \{\, a^2 \,\}$$

A few verifications:

$$a^2 \cdot b \quad = \quad a \cdot ab$$

$$a^2 \cdot aba \quad = \quad a^3 \cdot ba$$

$$aba \cdot a^2 \quad = \quad ab \cdot a^3$$

Centralizers: an example

Consider (from Choffrut-Karhumäki-Ollinger):

$$L \quad = \quad \{ a, a^3, b, ba, ab, aba \}$$

then a solution of the commutation equation is

$$X \quad = \quad L \cup \{ a^2 \}$$

A few verifications:

$$a^2 \cdot b \quad = \quad a \cdot ab$$

$$a^2 \cdot aba \quad = \quad a^3 \cdot ba$$

$$aba \cdot a^2 \quad = \quad ab \cdot a^3$$

Centralizers: an example

Consider (from Choffrut-Karhumäki-Ollinger):

$$L = \{\, a,\ a^3,\ b,\ ba,\ ab,\ aba \,\}$$

then a solution of the commutation equation is

$$X = L \cup \{\, a^2 \,\}$$

A few verifications:

$$a^2 \cdot b = a \cdot ab$$

$$a^2 \cdot aba = a^3 \cdot ba$$

$$aba \cdot a^2 = ab \cdot a^3$$

Centralizers: an example

Consider (from Choffrut-Karhumäki-Ollinger):

$$L \quad = \quad \{ \, a, \, a^3, \, b, \, ba, \, ab, \, aba \, \}$$

then a solution of the commutation equation is

$$X \quad = \quad L \cup \{ \, a^2 \, \}$$

A few verifications:

$$a^2 \cdot b \quad = \quad a \cdot ab$$

$$a^2 \cdot aba \quad = \quad a^3 \cdot ba$$

$$aba \cdot a^2 \quad = \quad ab \cdot a^3$$

Centralizers

$$X \cdot L \;=\; L \cdot X$$

always has solutions, among which obviously:

$$\emptyset \quad \text{and} \quad \{\epsilon\} \quad \text{and} \quad L \quad \text{and} \quad L^* \quad \text{and} \quad L^+$$

The union of two solutions is a solution: if

$$X_1 \cdot L \;=\; L \cdot X_1 \quad \text{and} \quad X_2 \cdot L \;=\; L \cdot X_2$$

then

$$(X_1 \cup X_2) \cdot L \;=\; L \cdot (X_1 \cup X_2)$$

Same for intersection.

Centralizers

$$X \cdot L \;=\; L \cdot X$$

always has solutions, among which obviously:

$$\emptyset \quad \text{and} \quad \{\epsilon\} \quad \text{and} \quad L \quad \text{and} \quad L^* \quad \text{and} \quad L^+$$

The union of two solutions is a solution: if

$$X_1 \cdot L \;=\; L \cdot X_1 \quad \text{and} \quad X_2 \cdot L \;=\; L \cdot X_2$$

then

$$(X_1 \cup X_2) \cdot L \;=\; L \cdot (X_1 \cup X_2)$$

Same for intersection.

Centralizers

$$X \cdot L \ = \ L \cdot X$$

has a greatest solution: the union of all solutions (we will see why later).

It is the centralizer of L, denoted $\mathcal{C}(L)$.

It contains ϵ. Denote $\mathcal{C}_+(L)$ the largest solution not containing ϵ.

Centralizers

Approximation results:

$$L^* \quad \subseteq \quad \mathcal{C}(L) \quad \subseteq \quad \mathit{Pref}(L^*) \cap \mathit{Suff}(L^*)$$

$$L^+ \quad \subseteq \quad \mathcal{C}_+(L) \quad \subseteq \quad \mathit{Pref}_+(L^+) \cap \mathit{Suff}_+(L^+)$$

Centralizers, game-theoretically

Centralizers have a natural interactive interpretation: consider a two-player game, starting on a word

$$u \in A^*$$

Adam adds a word $w \in L$ to a side of u, and reaches Eve's position:

$$u \cdot w \ \in \ A^* \cdot L$$

Eve answers by removing a word $v \in L$ on the other side, reaching:

$$v^{-1}\, u\, w \ \in \ L^{-1} \cdot A^* \cdot L \quad \subseteq \quad A^*$$

Adam plays again, and so on:

$$u \xrightarrow{Adam} u \cdot w \xrightarrow{Eve} v^{-1}uw \xrightarrow{Adam} sv^{-1}uw \xrightarrow{Eve} \cdots$$

Centralizers, game-theoretically

Centralizers have a natural interactive interpretation: consider a two-player game, starting on a word

$$u \in A^*$$

Adam adds a word $w \in L$ to a side of u, and reaches Eve's position:

$$u \cdot w \;\in\; A^* \cdot L$$

Eve answers by removing a word $v \in L$ on the other side, reaching:

$$v^{-1}\, u\, w \;\in\; L^{-1} \cdot A^* \cdot L \quad \subseteq \quad A^*$$

Adam plays again, and so on:

$$u \xrightarrow{Adam} u \cdot w \xrightarrow{Eve} v^{-1}uw \xrightarrow{Adam} sv^{-1}uw \xrightarrow{Eve} \cdots$$

Centralizers, game-theoretically

Centralizers have a natural interactive interpretation: consider a two-player game, starting on a word

$$u \in A^*$$

Adam adds a word $w \in L$ to a side of u, and reaches Eve's position:

$$u \cdot w \ \in \ A^* \cdot L$$

Eve answers by removing a word $v \in L$ on the other side, reaching:

$$v^{-1}\, u\, w \ \in \ L^{-1} \cdot A^* \cdot L \quad \subseteq \quad A^*$$

Adam plays again, and so on:

$$u \xrightarrow{Adam} u \cdot w \xrightarrow{Eve} v^{-1}uw \xrightarrow{Adam} sv^{-1}uw \xrightarrow{Eve} \cdots$$

Centralizers, game-theoretically

If Eve can not answer at some point, she looses.
If she can play forever, Adam looses.

Eve's winning plays explore the commutation orbit of the initial word:

$$u \in \mathcal{C}(L) \quad \text{iff Eve can win every play starting from it}$$

A key proposition on centralizers

The game-theoretic interpretation, rephrased:

Proposition

Given u, $v \in L$, suppose that

$$u \cdot x \;=\; y \cdot v \tag{2}$$

Then

$$x \in \mathcal{C}(L) \quad \Longleftrightarrow \quad y \in \mathcal{C}(L)$$

Note that (2) means that x and y can commute with one word of L, and that this assumption is one-sided.

Commutation game: examples

$$L = a^+ b$$

Does $b \in \mathcal{C}(L)$?

$$b \xrightarrow{A,g} a^2 b^2 \xrightarrow{E,d} \emptyset$$

No.

(Adam could play any word of L on the left side).

Commutation game: examples

$$L \;=\; a^+ b \,+\, b(a^2)^+$$

Does $bab \in \mathcal{C}(L)$?

$$bab \;\xrightarrow{A,g}\; ab^2ab$$

Commutation game: examples

$$L \quad = \quad a^+ b \; + \; b(a^2)^+$$

Does $bab \in \mathcal{C}(L)$?

$$bab \quad \xrightarrow{A,g} \quad ab^2 ab \quad \xrightarrow{E,d} \quad ab^2$$

Commutation game: examples

$$L \quad = \quad a^+ b \; + \; b(a^2)^+$$

Does $bab \in \mathcal{C}(L)$?

$$bab \quad \xrightarrow{A,g} \quad ab^2 ab \quad \xrightarrow{E,d} \quad ab^2 \quad \xrightarrow{A,g} \quad abab^2$$

Commutation game: examples

$$L \;=\; a^+b \;+\; b(a^2)^+$$

Does $bab \in \mathcal{C}(L)$?

$$bab \xrightarrow{A,g} ab^2ab \xrightarrow{E,d} ab^2 \xrightarrow{A,g} abab^2 \xrightarrow{E,d} \emptyset$$

No – but Adam could have been smarter.

Commutation game: examples

$$L \quad = \quad a^+ b \; + \; b(a^2)^+$$

Does $bab \in \mathcal{C}(L)$?

$$bab \quad \xrightarrow{A,d} \quad babw$$

(for any $w \in L$)

Commutation game: examples

$$L \quad = \quad a^+ b \; + \; b(a^2)^+$$

Does $bab \in \mathcal{C}(L)$?

$$bab \quad \xrightarrow{A,d} \quad babw \quad \xrightarrow{E,g} \quad \emptyset$$

Can Adam always win in a move?

Commutation game: examples

$$L \quad = \quad a^+ b \ + \ b(a^2)^+$$

Does $ba^2 b \in \mathcal{C}(L)$?

$$ba^2 b \quad \xrightarrow{A,g} \quad a^2 b^2 a^2 b$$

Commutation game: examples

$$L \;=\; a^+b \;+\; b(a^2)^+$$

Does $ba^2b \in \mathcal{C}(L)$?

$$ba^2b \quad \xrightarrow{A,g} \quad a^2b^2a^2b \quad \xrightarrow{E,d} \quad a^2b^2$$

Commutation game: examples

$$L \quad = \quad a^+ b \; + \; b(a^2)^+$$

Does $ba^2b \in \mathcal{C}(L)$?

$$ba^2b \quad \xrightarrow{A,g} \quad a^2b^2a^2b \quad \xrightarrow{E,d} \quad a^2b^2 \qquad \xrightarrow{A,g} \quad aba^2b^2$$

Commutation game: examples

$$L \;=\; a^+b \;+\; b(a^2)^+$$

Does $ba^2b \in \mathcal{C}(L)$?

$$ba^2b \xrightarrow{A,g} a^2b^2a^2b \xrightarrow{E,d} a^2b^2 \xrightarrow{A,g} aba^2b^2 \xrightarrow{E,d} \emptyset$$

Not always: ba^2b has both a suffix and a prefix in L, so that Eve can always play a move.

Centralizers

Conway's problem: if L is regular, what can be said of $\mathcal{C}(L)$?

Open problem for a long time; it seems that people expected some regularity. Until:

Theorem (Kunc 2006)

- *There exists a regular, star-free language L such that $\mathcal{C}(L)$, $\mathcal{C}_+(L)$ and $\mathcal{C}(L) \setminus \mathcal{C}_+(L)$ are not recursively enumerable.*
- *There exists a finite language L such that $\mathcal{C}(L)$ and $\mathcal{C}_+(L)$ are not recursively enumerable.*

In the second statement, nothing is said about $\mathcal{C}(L) \setminus \mathcal{C}_+(L)$.

Centralizers

Conway's problem: if L is regular, what can be said of $\mathcal{C}(L)$?

Open problem for a long time; it seems that people expected some regularity. Until:

Theorem (Kunc 2006)

- *There exists a regular, star-free language L such that $\mathcal{C}(L)$, $\mathcal{C}_+(L)$ and $\mathcal{C}(L) \setminus \mathcal{C}_+(L)$ are not recursively enumerable.*
- *There exists a finite language L such that $\mathcal{C}(L)$ and $\mathcal{C}_+(L)$ are not recursively enumerable.*

In the second statement, nothing is said about $\mathcal{C}(L) \setminus \mathcal{C}_+(L)$.

Centralizers

Theorem (Kunc 2006)

- *There exists a regular, star-free language L such that $\mathcal{C}(L)$, $\mathcal{C}_+(L)$ and $\mathcal{C}(L) \setminus \mathcal{C}_+(L)$ are not recursively enumerable.*
- *There exists a finite language L such that $\mathcal{C}(L)$ and $\mathcal{C}_+(L)$ are not recursively enumerable.*

In this talk:

- we describe the main elements of Kunc's proof,
- we rephrase it in an alternate model of computation,
- and we reveal an important key for understanding this theorem: centralizers are coinductive.

Elements of Kunc's proof

First step: encode the behaviour of a Turing-complete machine in $\mathcal{C}(L)$.

We can only build L...

Two dual purposes:

- add words to simulate the machine's transitions,
- add words to restrict the centralizer (it should only "simulate" the transitions of the machine)

Encoding Minsky machines

Kunc encodes Minsky machines:

- two counters storing integers,
- a finite set of states,
- increasal/decreasal of counters,
- conditional operation (does a counter store 0?)

They are Turing-complete.

Encoding Minsky machines

Typical configuration:

$$(q,\ i,\ j) \quad \in \quad Q \times \mathbb{N} \times \mathbb{N}$$

Transitions update the counters.

If q is a state increasing the first counter and going to q':

$$(q,\ i,\ j) \quad \longrightarrow \quad (q',\ i+1,\ j)$$

Encoding Minsky machines

Kunc designs L such that $\mathcal{C}(L)$ contains every word

$$a^{n+1} \; b \; \widehat{a}^{m+1} \; \widehat{d_q}^2$$

encoding a configuration

$$(q, \; n, \; m)$$

How do the encodings of configurations relate ?

Encoding Minsky machines

To simulate in $\mathcal{C}(L)$ the increasing transition

$$(q,\, i,\, j) \quad \longrightarrow \quad (q',\, i+1,\, j)$$

Kunc uses the "game-theoretic intepretation" to obtain:

$$\begin{array}{lll} & a^{n+1}\ b\ \widehat{a}^{m+1}\ \widehat{d}_q^2 & \in \mathcal{C}(L) \\ \Longleftrightarrow & a^{n+2}\ b\ \widehat{a}^{m+1}\ \widehat{d}_{q'}^2 & \in \mathcal{C}(L) \end{array}$$

Encoding Minsky machines

Indeed, start from

$$a^{n+1}\ b\ \widehat{a}^{m+1}\ \widehat{d}_q^2 \in A^*$$

Then

$$g_q \cdot a \cdot a^{n+1}\ b\ \widehat{a}^{m+1}\ \widehat{d}_q^2 \in L \cdot A^*$$

And

$$g_q \cdot a^{n+2}\ b\ \widehat{a}^{m+1}\ \widehat{d}_q \cdot \widehat{d}_q \in A^* \cdot L$$

So that, by the Proposition,

$$a^{n+1}\ b\ \widehat{a}^{m+1}\ \widehat{d}_q^2 \in \mathcal{C}(L) \iff g_q a^{n+2}\ b\ \widehat{a}^{m+1}\ \widehat{d}_q \in \mathcal{C}(L)$$

Encoding Minsky machines

Indeed, start from

$$a^{n+1}\ b\ \widehat{a}^{m+1}\ \widehat{d}_q^2 \in A^*$$

Then

$$g_q \cdot a \cdot a^{n+1}\ b\ \widehat{a}^{m+1}\ \widehat{d}_q^2 \in L \cdot A^*$$

And

$$g_q \cdot a^{n+2}\ b\ \widehat{a}^{m+1}\ \widehat{d}_q \cdot \widehat{d}_q \in A^* \cdot L$$

So that, by the Proposition,

$$a^{n+1}\ b\ \widehat{a}^{m+1}\ \widehat{d}_q^2 \in \mathcal{C}(L) \iff g_q a^{n+2}\ b\ \widehat{a}^{m+1}\ \widehat{d}_q \in \mathcal{C}(L)$$

Encoding Minsky machines

Indeed, start from

$$a^{n+1}\ b\ \widehat{a}^{m+1}\ \widehat{d}_q^2 \in A^*$$

Then

$$g_q \cdot a \cdot a^{n+1}\ b\ \widehat{a}^{m+1}\ \widehat{d}_q^2 \in L \cdot A^*$$

And

$$g_q \cdot a^{n+2}\ b\ \widehat{a}^{m+1}\ \widehat{d}_q \cdot \widehat{d}_q \in A^* \cdot L$$

So that, by the Proposition,

$$a^{n+1}\ b\ \widehat{a}^{m+1}\ \widehat{d}_q^2 \in \mathcal{C}(L) \iff g_q a^{n+2}\ b\ \widehat{a}^{m+1}\ \widehat{d}_q \in \mathcal{C}(L)$$

Encoding Minsky machines

Indeed, start from

$$a^{n+1}\ b\ \widehat{a}^{m+1}\ \widehat{d}_q^2 \in A^*$$

Then

$$g_q \cdot a \cdot a^{n+1}\ b\ \widehat{a}^{m+1}\ \widehat{d}_q^2 \in L \cdot A^*$$

And

$$g_q \cdot a^{n+2}\ b\ \widehat{a}^{m+1}\ \widehat{d}_q \cdot \widehat{d}_q \in A^* \cdot L$$

So that, by the Proposition,

$$a^{n+1}\ b\ \widehat{a}^{m+1}\ \widehat{d}_q^2 \in \mathcal{C}(L) \iff g_q a^{n+2}\ b\ \widehat{a}^{m+1}\ \widehat{d}_q \in \mathcal{C}(L)$$

Encoding Minsky machines

Then, from

$$g_q \cdot a^{n+2}\ b\ \widehat{a}^{m+1}\ \widehat{d}_q \in A^*$$

we obtain

$$e_q \cdot f_q \cdot g_q \cdot a^{n+2}\ b\ \widehat{a}^{m+1}\ \widehat{d}_q \in L \cdot A^*$$

and

$$e_q \cdot f_q \cdot g_q \cdot a^{n+2}\ b\ \widehat{a}^{m+1}\ \widehat{d}_q \in A^* \cdot L$$

So that

$$a^{n+1}\ b\ \widehat{a}^{m+1}\ \widehat{d}_q^2 \in \mathcal{C}(L) \iff e_q f_q g_q a^{n+2}\ b\ \widehat{a}^{m+1}\ \in \mathcal{C}(L)$$

Encoding Minsky machines

Then, from

$$g_q \cdot a^{n+2}\ b\ \widehat{a}^{m+1}\ \widehat{d}_q \in A^*$$

we obtain

$$e_q \cdot f_q \cdot g_q \cdot a^{n+2}\ b\ \widehat{a}^{m+1}\ \widehat{d}_q \in L \cdot A^*$$

and

$$e_q \cdot f_q \cdot g_q \cdot a^{n+2}\ b\ \widehat{a}^{m+1}\ \widehat{d}_q \in A^* \cdot L$$

So that

$$a^{n+1}\ b\ \widehat{a}^{m+1}\ \widehat{d}_q^2 \in \mathcal{C}(L) \iff e_q f_q g_q a^{n+2}\ b\ \widehat{a}^{m+1} \in \mathcal{C}(L)$$

Encoding Minsky machines

Then, from

$$g_q \cdot a^{n+2}\ b\ \widehat{a}^{m+1}\ \widehat{d}_q \in A^*$$

we obtain

$$e_q \cdot f_q \cdot g_q \cdot a^{n+2}\ b\ \widehat{a}^{m+1}\ \widehat{d}_q \in L \cdot A^*$$

and

$$e_q \cdot f_q \cdot g_q \cdot a^{n+2}\ b\ \widehat{a}^{m+1}\ \widehat{d}_q \in A^* \cdot L$$

So that

$$a^{n+1}\ b\ \widehat{a}^{m+1}\ \widehat{d}_q^2 \in \mathcal{C}(L) \iff e_q f_q g_q a^{n+2}\ b\ \widehat{a}^{m+1} \in \mathcal{C}(L)$$

Encoding Minsky machines

Then, from

$$g_q \cdot a^{n+2}\ b\ \widehat{a}^{m+1}\ \widehat{d}_q \in A^*$$

we obtain

$$e_q \cdot f_q \cdot g_q \cdot a^{n+2}\ b\ \widehat{a}^{m+1}\ \widehat{d}_q \in L \cdot A^*$$

and

$$e_q \cdot f_q \cdot g_q \cdot a^{n+2}\ b\ \widehat{a}^{m+1}\ \widehat{d}_q \in A^* \cdot L$$

So that

$$a^{n+1}\ b\ \widehat{a}^{m+1}\ \widehat{d}_q^2 \in \mathcal{C}(L) \iff e_q f_q g_q a^{n+2}\ b\ \widehat{a}^{m+1}\ \in \mathcal{C}(L)$$

Encoding Minsky machines

Then, from

$$e_q f_q g_q a^{n+2} \; b \; \widehat{a}^{m+1} \;\in A^*$$

we obtain

$$e_q f_q g_q a^{n+2} \; b \; \widehat{a}^{m+1} \widehat{d}_{q'} \in A^* \cdot L$$

and

$$e_q f_q g_q a^{n+2} \; b \; \widehat{a}^{m+1} \widehat{d}_{q'} \in L \cdot A^*$$

So that

$$a^{n+1} \; b \; \widehat{a}^{m+1} \; \widehat{d}_q^2 \in \mathcal{C}(L) \iff f_q g_q a^{n+2} \; b \; \widehat{a}^{m+1} \widehat{d}_{q'} \;\in \mathcal{C}(L)$$

Encoding Minsky machines

Then, from

$$e_q f_q g_q a^{n+2} \; b \; \widehat{a}^{m+1} \; \in A^*$$

we obtain

$$e_q f_q g_q a^{n+2} \; b \; \widehat{a}^{m+1} \widehat{d}_{q'} \in A^* \cdot L$$

and

$$e_q f_q g_q a^{n+2} \; b \; \widehat{a}^{m+1} \widehat{d}_{q'} \in L \cdot A^*$$

So that

$$a^{n+1} \; b \; \widehat{a}^{m+1} \; \widehat{d}_q^2 \in \mathcal{C}(L) \iff f_q g_q a^{n+2} \; b \; \widehat{a}^{m+1} \widehat{d}_{q'} \; \in \mathcal{C}(L)$$

Encoding Minsky machines

Then, from

$$e_q f_q g_q a^{n+2}\ b\ \widehat{a}^{m+1}\ \in A^*$$

we obtain

$$e_q f_q g_q a^{n+2}\ b\ \widehat{a}^{m+1} \widehat{d}_{q'} \in A^* \cdot L$$

and

$$e_q f_q g_q a^{n+2}\ b\ \widehat{a}^{m+1} \widehat{d}_{q'} \in L \cdot A^*$$

So that

$$a^{n+1}\ b\ \widehat{a}^{m+1}\ \widehat{d}_q^2 \in \mathcal{C}(L) \iff f_q g_q a^{n+2}\ b\ \widehat{a}^{m+1} \widehat{d}_{q'}\ \in \mathcal{C}(L)$$

Encoding Minsky machines

Then, from

$$e_q f_q g_q a^{n+2}\ b\ \widehat{a}^{m+1}\ \in A^*$$

we obtain

$$e_q f_q g_q a^{n+2}\ b\ \widehat{a}^{m+1} \widehat{d}_{q'} \in A^* \cdot L$$

and

$$e_q f_q g_q a^{n+2}\ b\ \widehat{a}^{m+1} \widehat{d}_{q'} \in L \cdot A^*$$

So that

$$a^{n+1}\ b\ \widehat{a}^{m+1}\ \widehat{d}_q^2 \in \mathcal{C}(L) \iff f_q g_q a^{n+2}\ b\ \widehat{a}^{m+1} \widehat{d}_{q'}\ \in \mathcal{C}(L)$$

Encoding Minsky machines

Finally, from

$$f_q g_q a^{n+2}\ b\ \widehat{a}^{m+1}\widehat{d}_{q'}\ \in A^*$$

we obtain

$$f_q g_q a^{n+2}\ b\ \widehat{a}^{m+1}\widehat{d}_{q'} \cdot \widehat{d}_{q'}\ \in A^* \cdot L$$

and

$$f_q g_q \cdot a^{n+2}\ b\ \widehat{a}^{m+1}\widehat{d}_{q'} \cdot \widehat{d}_{q'}\ \in L \cdot A^*$$

So that we related two configurations of the machine:

$$a^{n+1}\ b\ \widehat{a}^{m+1}\ \widehat{d}_q^2 \in \mathcal{C}(L) \iff a^{n+2}\ b\ \widehat{a}^{m+1}\widehat{d}_{q'}^2\ \in \mathcal{C}(L)$$

L is defined such that only valid transitions can be simulated in $\mathcal{C}(L)$.

Encoding Minsky machines

Finally, from

$$f_q g_q a^{n+2}\ b\ \widehat{a}^{m+1}\widehat{d}_{q'}\ \in A^*$$

we obtain

$$f_q g_q a^{n+2}\ b\ \widehat{a}^{m+1}\widehat{d}_{q'} \cdot \widehat{d}_{q'}\ \in A^* \cdot L$$

and

$$f_q g_q \cdot a^{n+2}\ b\ \widehat{a}^{m+1}\widehat{d}_{q'} \cdot \widehat{d}_{q'}\ \in L \cdot A^*$$

So that we related two configurations of the machine:

$$a^{n+1}\ b\ \widehat{a}^{m+1}\ \widehat{d}_q^2 \in \mathcal{C}(L) \iff a^{n+2}\ b\ \widehat{a}^{m+1}\widehat{d}_{q'}^2\ \in \mathcal{C}(L)$$

L is defined such that only valid transitions can be simulated in $\mathcal{C}(L)$.

Encoding Minsky machines

Finally, from

$$f_q g_q a^{n+2}\ b\ \widehat{a}^{m+1} \widehat{d}_{q'}\ \in A^*$$

we obtain

$$f_q g_q a^{n+2}\ b\ \widehat{a}^{m+1} \widehat{d}_{q'} \cdot \widehat{d}_{q'}\ \in A^* \cdot L$$

and

$$f_q g_q \cdot a^{n+2}\ b\ \widehat{a}^{m+1} \widehat{d}_{q'} \cdot \widehat{d}_{q'}\ \in L \cdot A^*$$

So that we related two configurations of the machine:

$$a^{n+1}\ b\ \widehat{a}^{m+1}\ \widehat{d}_q^2 \in \mathcal{C}(L) \iff a^{n+2}\ b\ \widehat{a}^{m+1} \widehat{d}_{q'}^2\ \in \mathcal{C}(L)$$

L is defined such that only valid transitions can be simulated in $\mathcal{C}(L)$.

Encoding Minsky machines

Finally, from

$$f_q g_q a^{n+2}\ b\ \widehat{a}^{m+1} \widehat{d}_{q'}\ \in A^*$$

we obtain

$$f_q g_q a^{n+2}\ b\ \widehat{a}^{m+1} \widehat{d}_{q'} \cdot \widehat{d}_{q'}\ \in A^* \cdot L$$

and

$$f_q g_q \cdot a^{n+2}\ b\ \widehat{a}^{m+1} \widehat{d}_{q'} \cdot \widehat{d}_{q'}\ \in L \cdot A^*$$

So that we related two configurations of the machine:

$$a^{n+1}\ b\ \widehat{a}^{m+1}\ \widehat{d}_q^2 \in \mathcal{C}(L) \iff a^{n+2}\ b\ \widehat{a}^{m+1} \widehat{d}_{q'}^2\ \in \mathcal{C}(L)$$

L is defined such that only valid transitions can be simulated in $\mathcal{C}(L)$.

Encoding Minsky machines: a summary

$$
\begin{array}{rrl}
 & a^{n+1}\ b\ \widehat{a}^{m+1}\ \widehat{d}_q^2 & \in \mathcal{C}(L) \\
\Longleftrightarrow & g_q\ a^{n+2}\ b\ \widehat{a}^{m+1}\ \widehat{d}_q & \in \mathcal{C}(L) \\
\Longleftrightarrow & e_q\ f_q\ g_q\ a^{n+2}\ b\ \widehat{a}^{m+1} & \in \mathcal{C}(L) \\
\Longleftrightarrow & f_q\ g_q\ a^{n+2}\ b\ \widehat{a}^{m+1}\ \widehat{d}_{q'} & \in \mathcal{C}(L) \\
\Longleftrightarrow & a^{n+2}\ b\ \widehat{a}^{m+1}\ \widehat{d}_{q'}^2 & \in \mathcal{C}(L)
\end{array}
$$

$e_q,\ f_q g_q,\ \ldots \in L$ allow to manipulate data: add or remove letters, and carry state information.

$\widehat{d}_q$ does not affect data, but updates the state information.

Encoding Minsky machines: a summary

$$
\begin{array}{rrl}
 & a^{n+1}\ b\ \widehat{a}^{m+1}\ \widehat{d}_q^2 & \in \mathcal{C}(L) \\
\Longleftrightarrow & g_q\ a^{n+2}\ b\ \widehat{a}^{m+1}\ \widehat{d}_q & \in \mathcal{C}(L) \\
\Longleftrightarrow & e_q\ f_q\ g_q\ a^{n+2}\ b\ \widehat{a}^{m+1} & \in \mathcal{C}(L) \\
\Longleftrightarrow & f_q\ g_q\ a^{n+2}\ b\ \widehat{a}^{m+1}\ \widehat{d}_{q'} & \in \mathcal{C}(L) \\
\Longleftrightarrow & a^{n+2}\ b\ \widehat{a}^{m+1}\ \widehat{d}_{q'}^2 & \in \mathcal{C}(L)
\end{array}
$$

$e_q,\ f_q g_q,\ \ldots \in L$ allow to manipulate data: add or remove letters, and carry state information.

$\widehat{d}_q$ does not affect data, but updates the state information.

Encoding Minsky machines: a summary

$$
\begin{array}{rrl}
 & a^{n+1}\ b\ \widehat{a}^{m+1}\ \widehat{d}_q^2 & \in \mathcal{C}(L) \\
\Longleftrightarrow & g_q\ a^{n+2}\ b\ \widehat{a}^{m+1}\ \widehat{d}_q & \in \mathcal{C}(L) \\
\Longleftrightarrow & e_q\ f_q\ g_q\ a^{n+2}\ b\ \widehat{a}^{m+1} & \in \mathcal{C}(L) \\
\Longleftrightarrow & f_q\ g_q\ a^{n+2}\ b\ \widehat{a}^{m+1}\ \widehat{d}_{q'} & \in \mathcal{C}(L) \\
\Longleftrightarrow & a^{n+2}\ b\ \widehat{a}^{m+1}\ \widehat{d}_{q'}^2 & \in \mathcal{C}(L)
\end{array}
$$

$e_q,\ f_q g_q,\ \ldots \in L$ allow to manipulate data: add or remove letters, and carry state information.

$\widehat{d}_q$ does not affect data, but updates the state information.

Encoding Minsky machines: a summary

$$
\begin{array}{rrl}
 & a^{n+1}\ b\ \widehat{a}^{m+1}\ \widehat{d}_q^2 & \in \mathcal{C}(L) \\
\Longleftrightarrow & g_q\ a^{n+2}\ b\ \widehat{a}^{m+1}\ \widehat{d}_q & \in \mathcal{C}(L) \\
\Longleftrightarrow & e_q\ f_q\ g_q\ a^{n+2}\ b\ \widehat{a}^{m+1} & \in \mathcal{C}(L) \\
\Longleftrightarrow & f_q\ g_q\ a^{n+2}\ b\ \widehat{a}^{m+1}\ \widehat{d}_{q'} & \in \mathcal{C}(L) \\
\Longleftrightarrow & a^{n+2}\ b\ \widehat{a}^{m+1}\ \widehat{d}_{q'}^2 & \in \mathcal{C}(L)
\end{array}
$$

$e_q,\ f_q g_q,\ \ldots \in L$ allow to manipulate data: add or remove letters, and carry state information.

$\widehat{d}_q$ does not affect data, but updates the state information.

Encoding Minsky machines: a summary

$$
\begin{array}{rrl}
 & a^{n+1}\ b\ \widehat{a}^{m+1}\ \widehat{d}_q^2 & \in \mathcal{C}(L) \\
\Longleftrightarrow & g_q\ a^{n+2}\ b\ \widehat{a}^{m+1}\ \widehat{d}_q & \in \mathcal{C}(L) \\
\Longleftrightarrow & e_q\ f_q\ g_q\ a^{n+2}\ b\ \widehat{a}^{m+1} & \in \mathcal{C}(L) \\
\Longleftrightarrow & f_q\ g_q\ a^{n+2}\ b\ \widehat{a}^{m+1}\ \widehat{d}_{q'} & \in \mathcal{C}(L) \\
\Longleftrightarrow & a^{n+2}\ b\ \widehat{a}^{m+1}\ \widehat{d}_{q'}^2 & \in \mathcal{C}(L)
\end{array}
$$

$e_q,\ f_q g_q,\ \ldots \in L$ allow to manipulate data: add or remove letters, and carry state information.

$\widehat{d}_q$ does not affect data, but updates the state information.

Encoding Minsky machines: a summary

$$\begin{array}{rrl}
 & a^{n+1}\ b\ \widehat{a}^{m+1}\ \widehat{d}_q^2 & \in \mathcal{C}(L) \\
\Longleftrightarrow & g_q\ a^{n+2}\ b\ \widehat{a}^{m+1}\ \widehat{d}_q & \in \mathcal{C}(L) \\
\Longleftrightarrow & e_q\ f_q\ g_q\ a^{n+2}\ b\ \widehat{a}^{m+1} & \in \mathcal{C}(L) \\
\Longleftrightarrow & f_q\ g_q\ a^{n+2}\ b\ \widehat{a}^{m+1}\ \widehat{d}_{q'} & \in \mathcal{C}(L) \\
\Longleftrightarrow & a^{n+2}\ b\ \widehat{a}^{m+1}\ \widehat{d}_{q'}^2 & \in \mathcal{C}(L)
\end{array}$$

$e_q,\ f_q g_q,\ \ldots \in L$ allow to manipulate data: add or remove letters, and carry state information.

$\widehat{d}_q$ does not affect data, but updates the state information.

Encoding Minsky machines: a summary

Similar encoding of decreasal/counter testing, for both counters.

Second counter: symmetry and use of:

$$a^{n+1}\ b\ \widehat{a}^{m+1}\ \widehat{d_q^2} \in \mathcal{C}(L) \quad \Longleftrightarrow \quad d_q^2\ a^{n+1}\ b\ \widehat{a}^{m+1} \in \mathcal{C}(L)$$

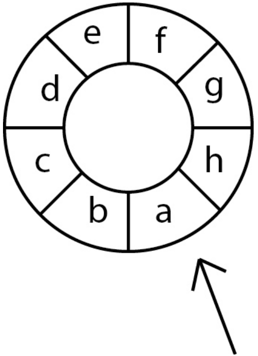
Minsky machines have too many operations. Let's use a simpler model!

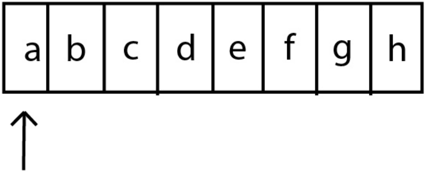
Clockwise Turing machines

Clockwise Turing machines (Neary -Woods) have only one kind of transition.

- one circular tape,
- a clockwise-moving head,
- can output two symbols at once to extend the tape.

Clockwise Turing machines vs. Turing machines

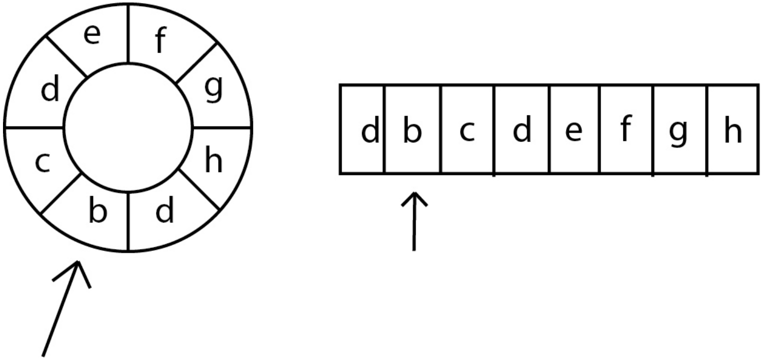




Both machines in state q.

The Turing machine reads a, writes d and moves head to the right...

Clockwise Turing machines vs. Turing machines



where both machines have state q'.

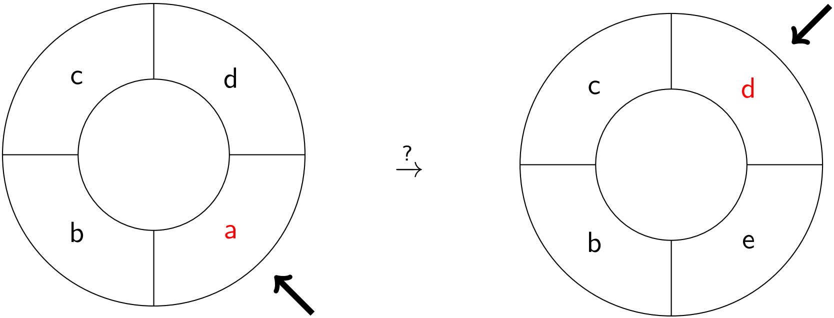
Clockwise Turing machines

Clockwise Turing machines simulate Turing machines.

No need to store the size of the tape to simulate a counter-clockwise transition. Just need about $|Q| \times |\Sigma|$ more states.

(crucial! encoding an unbounded register $\Rightarrow$ infinite alphabet)

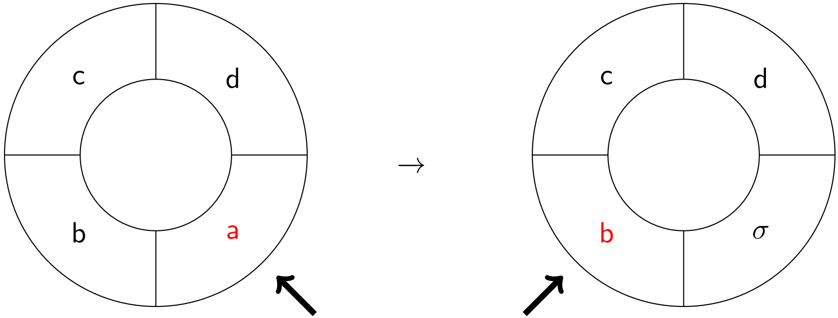
Clockwise Turing machines



Head on a.

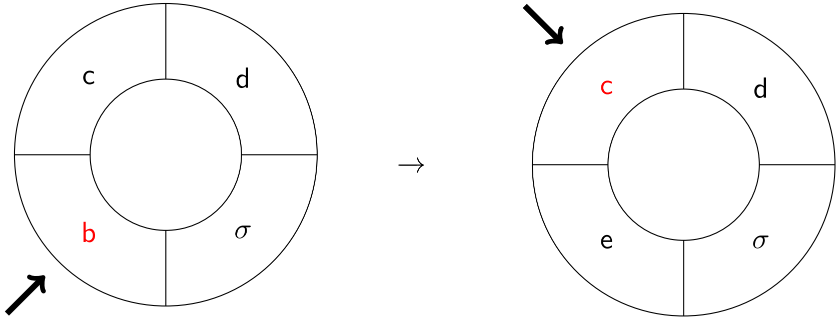
We want to write e and move counter-clockwise.

Clockwise Turing machines



a is replaced with a special symbol σ.
The state "remembers" that *e* needs to be translated.
The head moves clockwise.

Clockwise Turing machines

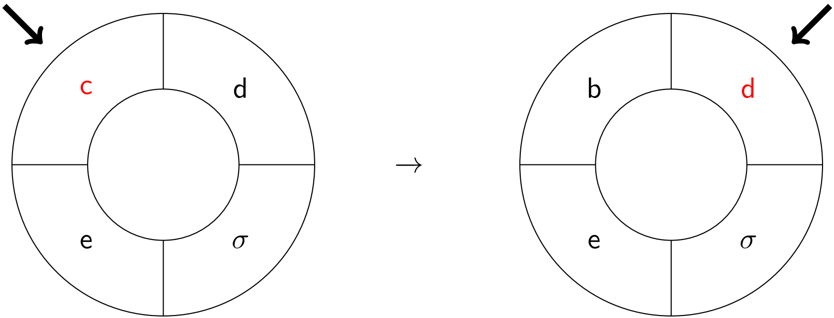


b is replaced with e.

The state "remembers" that b needs to be translated.

The head moves clockwise.

Clockwise Turing machines

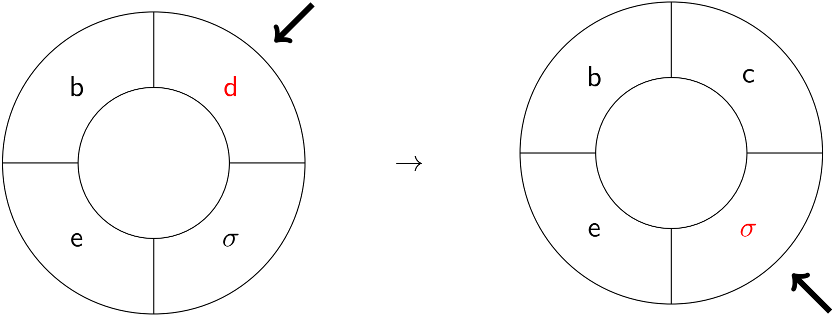


c is replaced with b.

The state "remembers" that c needs to be translated.

The head moves clockwise.

Clockwise Turing machines

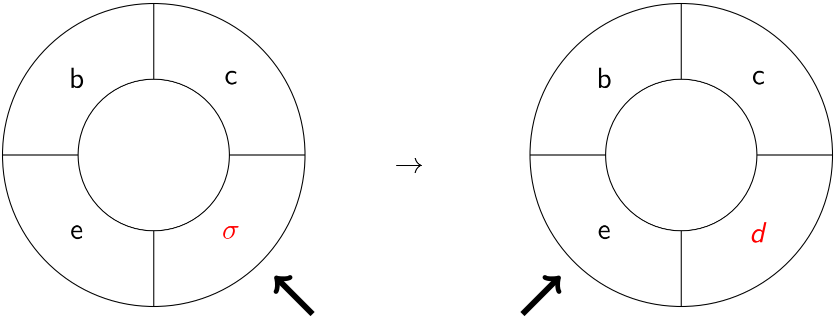


d is replaced with c.

The state "remembers" that d needs to be translated.

The head moves clockwise.

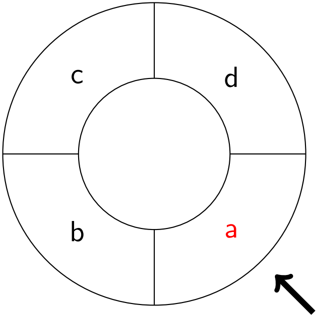
Clockwise Turing machines



σ is replaced with d.
The state changes to q'.
The head moves clockwise.

Simulation is performed, up to a harmless rotation.

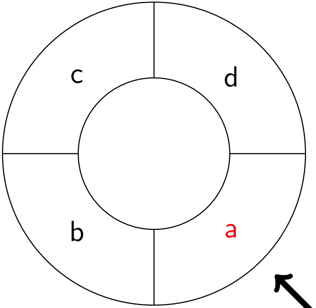
Clockwise Turing machines: encoding configurations



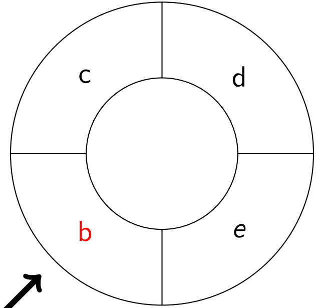
in state q is encoded as the word

$$a\ b\ c\ d\ \widehat{d}_q^2$$

Clockwise Turing machines: encoding configurations



$\rightarrow$



from state q to state q' will be encoded as

$$a\ b\ c\ d\ \widehat{d}_q^2 \in \mathcal{C}(L) \iff b\ c\ d\ e\ \widehat{d}_{q'}^2 \in \mathcal{C}(L)$$

Clockwise Turing machines

We can use Kunc's ideas to define L such that a transition

$$\delta(q, u_1) \;=\; (v, q')$$

when the circular tape contains $u_1 \cdots u_n$ corresponds to:

$$\begin{array}{rrr}
 & u_1\ u_2\ \cdots\ u_n \widehat{d_q^2} & \in \mathcal{C}(L) \\
\Longleftrightarrow & f_{q,u_1}\ g_{q,u_1}\ u_1\ u_2\ \cdots\ u_n\ d_q & \in \mathcal{C}(L) \\
\Longleftrightarrow & e_{q,u_1}\ f_{q,u_1}\ g_{q,u_1}\ u_1\ u_2\ \cdots\ u_n & \in \mathcal{C}(L) \\
\Longleftrightarrow & g_{q,u_1}\ u_1\ u_2\ \cdots\ u_n\ v\ \widehat{g}_{q',v} & \in \mathcal{C}(L) \\
\Longleftrightarrow & u_2\ \cdots\ u_n\ v\ \widehat{g}_{q',v}\ \widehat{f}_{q',v}\ \widehat{e}_{q',v} & \in \mathcal{C}(L) \\
\Longleftrightarrow & d_{q'}\ u_2\ \cdots\ u_n\ v\ \widehat{g}_{q',v}\ \widehat{f}_{q',v} & \in \mathcal{C}(L) \\
\Longleftrightarrow & d_{q'}^2\ u_2\ \cdots\ u_n\ v & \in \mathcal{C}(L) \\
\Longleftrightarrow & u_2\ \cdots\ u_n\ v\ \widehat{d_{q'}^2} & \in \mathcal{C}(L)
\end{array}$$

Clockwise Turing machines

We can use Kunc's ideas to define L such that a transition

$$\delta(q, u_1) \;=\; (v, q')$$

when the circular tape contains $u_1 \cdots u_n$ corresponds to:

$$\begin{array}{lcr}
 & u_1\ u_2\ \cdots\ u_n \widehat{d_q^2} & \in \mathcal{C}(L) \\
\Longleftrightarrow & f_{q,u_1}\ g_{q,u_1}\ u_1\ u_2\ \cdots\ u_n\ \widehat{d_q} & \in \mathcal{C}(L) \\
\Longleftrightarrow & e_{q,u_1}\ f_{q,u_1}\ g_{q,u_1}\ u_1\ u_2\ \cdots\ u_n & \in \mathcal{C}(L) \\
\Longleftrightarrow & g_{q,u_1}\ u_1\ u_2\ \cdots\ u_n\ v\ \widehat{g}_{q',v} & \in \mathcal{C}(L) \\
\Longleftrightarrow & u_2\ \cdots\ u_n\ v\ \widehat{g}_{q',v}\ \widehat{f}_{q',v}\ \widehat{e}_{q',v} & \in \mathcal{C}(L) \\
\Longleftrightarrow & d_{q'}\ u_2\ \cdots\ u_n\ v\ \widehat{g}_{q',v}\ \widehat{f}_{q',v} & \in \mathcal{C}(L) \\
\Longleftrightarrow & d_{q'}^2\ u_2\ \cdots\ u_n\ v & \in \mathcal{C}(L) \\
\Longleftrightarrow & u_2\ \cdots\ u_n\ v\ \widehat{d_{q'}^2} & \in \mathcal{C}(L)
\end{array}$$

Clockwise Turing machines

We can use Kunc's ideas to define L such that a transition

$$\delta(q, u_1) \; = \; (v, q')$$

when the circular tape contains $u_1 \cdots u_n$ corresponds to:

$$
\begin{array}{rrl}
 & u_1 \; u_2 \; \cdots \; u_n \widehat{d_q^2} & \in \mathcal{C}(L) \\
\Longleftrightarrow & f_{q,u_1} \; g_{q,u_1} \; u_1 \; u_2 \; \cdots \; u_n \; \widehat{d_q} & \in \mathcal{C}(L) \\
\Longleftrightarrow & e_{q,u_1} \; f_{q,u_1} \; g_{q,u_1} \; u_1 \; u_2 \; \cdots \; u_n & \in \mathcal{C}(L) \\
\Longleftrightarrow & g_{q,u_1} \; u_1 \; u_2 \; \cdots \; u_n \; v \; \widehat{g}_{q',v} & \in \mathcal{C}(L) \\
\Longleftrightarrow & u_2 \; \cdots \; u_n \; v \; \widehat{g}_{q',v} \; \widehat{f}_{q',v} \; \widehat{e}_{q',v} & \in \mathcal{C}(L) \\
\Longleftrightarrow & d_{q'} \; u_2 \; \cdots \; u_n \; v \; \widehat{g}_{q',v} \; \widehat{f}_{q',v} & \in \mathcal{C}(L) \\
\Longleftrightarrow & d_{q'}^2 \; u_2 \; \cdots \; u_n \; v & \in \mathcal{C}(L) \\
\Longleftrightarrow & u_2 \; \cdots \; u_n \; v \; \widehat{d_{q'}^2} & \in \mathcal{C}(L)
\end{array}
$$

Clockwise Turing machines

We can use Kunc's ideas to define L such that a transition

$$\delta(q, u_1) = (v, q')$$

when the circular tape contains $u_1 \cdots u_n$ corresponds to:

$$\begin{array}{rcr}
 & u_1 \; u_2 \; \cdots \; u_n \widehat{d_q^2} & \in \mathcal{C}(L) \\
\Longleftrightarrow & f_{q,u_1} \; g_{q,u_1} \; u_1 \; u_2 \; \cdots \; u_n \; d_q & \in \mathcal{C}(L) \\
\Longleftrightarrow & e_{q,u_1} \; f_{q,u_1} \; g_{q,u_1} \; u_1 \; u_2 \; \cdots \; u_n & \in \mathcal{C}(L) \\
\Longleftrightarrow & g_{q,u_1} \; u_1 \; u_2 \; \cdots \; u_n \; v \; \widehat{g}_{q',v} & \in \mathcal{C}(L) \\
\Longleftrightarrow & u_2 \; \cdots \; u_n \; v \; \widehat{g}_{q',v} \; \widehat{f}_{q',v} \; \widehat{e}_{q',v} & \in \mathcal{C}(L) \\
\Longleftrightarrow & d_{q'} \; u_2 \; \cdots \; u_n \; v \; \widehat{g}_{q',v} \; \widehat{f}_{q',v} & \in \mathcal{C}(L) \\
\Longleftrightarrow & d_{q'}^2 \; u_2 \; \cdots \; u_n \; v & \in \mathcal{C}(L) \\
\Longleftrightarrow & u_2 \; \cdots \; u_n \; v \; \widehat{d_{q'}^2} & \in \mathcal{C}(L)
\end{array}$$

Clockwise Turing machines

We can use Kunc's ideas to define L such that a transition

$$\delta(q, u_1) = (v, q')$$

when the circular tape contains $u_1 \cdots u_n$ corresponds to:

$$\begin{array}{rll}
 & u_1\ u_2\ \cdots\ u_n \widehat{d_q^2} & \in \mathcal{C}(L) \\
\Longleftrightarrow & f_{q,u_1}\ g_{q,u_1}\ u_1\ u_2\ \cdots\ u_n\ d_q & \in \mathcal{C}(L) \\
\Longleftrightarrow & e_{q,u_1}\ f_{q,u_1}\ g_{q,u_1}\ u_1\ u_2\ \cdots\ u_n & \in \mathcal{C}(L) \\
\Longleftrightarrow & g_{q,u_1}\ u_1\ u_2\ \cdots\ u_n\ v\ \widehat{g}_{q',v} & \in \mathcal{C}(L) \\
\Longleftrightarrow & u_2\ \cdots\ u_n\ v\ \widehat{g}_{q',v}\ \widehat{f}_{q',v}\ \widehat{e}_{q',v} & \in \mathcal{C}(L) \\
\Longleftrightarrow & d_{q'}\ u_2\ \cdots\ u_n\ v\ \widehat{g}_{q',v}\ \widehat{f}_{q',v} & \in \mathcal{C}(L) \\
\Longleftrightarrow & d_{q'}^2\ u_2\ \cdots\ u_n\ v & \in \mathcal{C}(L) \\
\Longleftrightarrow & u_2\ \cdots\ u_n\ v\ \widehat{d_{q'}^2} & \in \mathcal{C}(L)
\end{array}$$

Clockwise Turing machines

We can use Kunc's ideas to define L such that a transition

$$\delta(q, u_1) \; = \; (v, q')$$

when the circular tape contains $u_1 \cdots u_n$ corresponds to:

$$\begin{array}{rrr}
 & u_1 \; u_2 \; \cdots \; u_n \widehat{d_q^2} & \in \mathcal{C}(L) \\
\Longleftrightarrow & f_{q,u_1} \; g_{q,u_1} \; u_1 \; u_2 \; \cdots \; u_n \; \widehat{d_q} & \in \mathcal{C}(L) \\
\Longleftrightarrow & e_{q,u_1} \; f_{q,u_1} \; g_{q,u_1} \; u_1 \; u_2 \; \cdots \; u_n & \in \mathcal{C}(L) \\
\Longleftrightarrow & g_{q,u_1} \; u_1 \; u_2 \; \cdots \; u_n \; v \; \widehat{g}_{q',v} & \in \mathcal{C}(L) \\
\Longleftrightarrow & u_2 \; \cdots \; u_n \; v \; \widehat{g}_{q',v} \; \widehat{f}_{q',v} \; \widehat{e}_{q',v} & \in \mathcal{C}(L) \\
\Longleftrightarrow & d_{q'} \; u_2 \; \cdots \; u_n \; v \; \widehat{g}_{q',v} \; \widehat{f}_{q',v} & \in \mathcal{C}(L) \\
\Longleftrightarrow & d_{q'}^2 \; u_2 \; \cdots \; u_n \; v & \in \mathcal{C}(L) \\
\Longleftrightarrow & u_2 \; \cdots \; u_n \; v \; \widehat{d_{q'}^2} & \in \mathcal{C}(L)
\end{array}$$

Clockwise Turing machines

We can use Kunc's ideas to define L such that a transition

$$\delta(q, u_1) \;=\; (v, q')$$

when the circular tape contains $u_1 \cdots u_n$ corresponds to:

$$\begin{array}{rrl}
 & u_1 \; u_2 \; \cdots \; u_n \widehat{d_q^2} & \in \mathcal{C}(L) \\
\Longleftrightarrow & f_{q,u_1} \; g_{q,u_1} \; u_1 \; u_2 \; \cdots \; u_n \; \widehat{d_q} & \in \mathcal{C}(L) \\
\Longleftrightarrow & e_{q,u_1} \; f_{q,u_1} \; g_{q,u_1} \; u_1 \; u_2 \; \cdots \; u_n & \in \mathcal{C}(L) \\
\Longleftrightarrow & g_{q,u_1} \; u_1 \; u_2 \; \cdots \; u_n \; v \; \widehat{g}_{q',v} & \in \mathcal{C}(L) \\
\Longleftrightarrow & u_2 \; \cdots \; u_n \; v \; \widehat{g}_{q',v} \; \widehat{f}_{q',v} \; \widehat{e}_{q',v} & \in \mathcal{C}(L) \\
\Longleftrightarrow & d_{q'} \; u_2 \; \cdots \; u_n \; v \; \widehat{g}_{q',v} \; \widehat{f}_{q',v} & \in \mathcal{C}(L) \\
\Longleftrightarrow & d_{q'}^2 \; u_2 \; \cdots \; u_n \; v & \in \mathcal{C}(L) \\
\Longleftrightarrow & u_2 \; \cdots \; u_n \; v \; \widehat{d_{q'}^2} & \in \mathcal{C}(L)
\end{array}$$

Clockwise Turing machines

We can use Kunc's ideas to define L such that a transition

$$\delta(q, u_1) = (v, q')$$

when the circular tape contains $u_1 \cdots u_n$ corresponds to:

$$\begin{array}{rcr}
 & u_1\ u_2\ \cdots\ u_n \widehat{d_q^2} & \in \mathcal{C}(L) \\
\Longleftrightarrow & f_{q,u_1}\ g_{q,u_1}\ u_1\ u_2\ \cdots\ u_n\ \widehat{d_q} & \in \mathcal{C}(L) \\
\Longleftrightarrow & e_{q,u_1}\ f_{q,u_1}\ g_{q,u_1}\ u_1\ u_2\ \cdots\ u_n & \in \mathcal{C}(L) \\
\Longleftrightarrow & g_{q,u_1}\ u_1\ u_2\ \cdots\ u_n\ v\ \widehat{g}_{q',v} & \in \mathcal{C}(L) \\
\Longleftrightarrow & u_2\ \cdots\ u_n\ v\ \widehat{g}_{q',v}\ \widehat{f}_{q',v}\ \widehat{e}_{q',v} & \in \mathcal{C}(L) \\
\Longleftrightarrow & d_{q'}\ u_2\ \cdots\ u_n\ v\ \widehat{g}_{q',v}\ \widehat{f}_{q',v} & \in \mathcal{C}(L) \\
\Longleftrightarrow & d_{q'}^2\ u_2\ \cdots\ u_n\ v & \in \mathcal{C}(L) \\
\Longleftrightarrow & u_2\ \cdots\ u_n\ v\ \widehat{d_{q'}^2} & \in \mathcal{C}(L)
\end{array}$$

Clockwise Turing machines

We can use Kunc's ideas to define L such that a transition

$$\delta(q, u_1) \;=\; (v, q')$$

when the circular tape contains $u_1 \cdots u_n$ corresponds to:

$$\begin{array}{rrl}
 & u_1\ u_2\ \cdots\ u_n \widehat{d_q^2} & \in \mathcal{C}(L) \\
\Longleftrightarrow & f_{q,u_1}\ g_{q,u_1}\ u_1\ u_2\ \cdots\ u_n\ \widehat{d_q} & \in \mathcal{C}(L) \\
\Longleftrightarrow & e_{q,u_1}\ f_{q,u_1}\ g_{q,u_1}\ u_1\ u_2\ \cdots\ u_n & \in \mathcal{C}(L) \\
\Longleftrightarrow & g_{q,u_1}\ u_1\ u_2\ \cdots\ u_n\ v\ \widehat{g}_{q',v} & \in \mathcal{C}(L) \\
\Longleftrightarrow & u_2\ \cdots\ u_n\ v\ \widehat{g}_{q',v}\ \widehat{f}_{q',v}\ \widehat{e}_{q',v} & \in \mathcal{C}(L) \\
\Longleftrightarrow & d_{q'}\ u_2\ \cdots\ u_n\ v\ \widehat{g}_{q',v}\ \widehat{f}_{q',v} & \in \mathcal{C}(L) \\
\Longleftrightarrow & d_{q'}^2\ u_2\ \cdots\ u_n\ v & \in \mathcal{C}(L) \\
\Longleftrightarrow & u_2\ \cdots\ u_n\ v\ \widehat{d_{q'}^2} & \in \mathcal{C}(L)
\end{array}$$

Elements of the centralizer

We can prove that the encoding of every configuration is in $\mathcal{C}(L)$.

As in Kunc's proof: check by hand that the set of encodings commutes with L.

Recursive enumerability

With this encoding, we intuitively get that centralizers can encode recursively enumerable languages, as they simulate the behaviour of Turing machines.

But where does the non-r.e. comes from ?

Intuition is somehow misleading, because centralizers are coinductive.

As such, they compute the whole configuration graph of the machine.

Another key ingredient of Kunc's proof is to remove encodings of the initial configurations of $\mathcal{C}(L)$: what remains corresponds to the complementary of the language of the encoded machine.

Recursive enumerability

With this encoding, we intuitively get that centralizers can encode recursively enumerable languages, as they simulate the behaviour of Turing machines.

But where does the non-r.e. comes from ?

Intuition is somehow misleading, because centralizers are coinductive.

As such, they compute the whole configuration graph of the machine.

Another key ingredient of Kunc's proof is to remove encodings of the initial configurations of $\mathcal{C}(L)$: what remains corresponds to the complementary of the language of the encoded machine.

Recursive enumerability

With this encoding, we intuitively get that centralizers can encode recursively enumerable languages, as they simulate the behaviour of Turing machines.

But where does the non-r.e. comes from ?

Intuition is somehow misleading, because centralizers are coinductive.

As such, they compute the whole configuration graph of the machine.

Another key ingredient of Kunc's proof is to remove encodings of the initial configurations of $\mathcal{C}(L)$: what remains corresponds to the complementary of the language of the encoded machine.

Induction vs. coinduction

Inductive construction:

- start from some initial element
- iterate a construction over it.

Point of view of calculus: a machine starts on an initial configuration and iterates its transition function over it.

Inductive interpretations only build finitary objects $\rightarrow$ terminating computations.

Induction vs. coinduction

Coinductive construction:

- start from all elements
- iterate a destruction over it (remove the elements contradicting some construction/deduction rule).

For calculus, this corresponds to the configuration graph of a machine:

1. Start with the (countable) complete graph whose vertices are the configurations:
$$V \quad = \quad A^* \times Q$$
2. Iteratively remove the edges which do not correspond to a transition of the machine.

Induction vs. coinduction: lattices and fixed points

Theorem (Tarski-Knaster)

Let $\mathcal{L}$ be a complete lattice and let

$$f \;:\; \mathcal{L} \longrightarrow \mathcal{L}$$

be an order-preserving function.
Then the set of fixed points of f in $\mathcal{L}$ is also a complete lattice.

In other terms: if you define a function f on an ordered structure with supremum, infimum, least and greatest element, then it has fixed points.

Moreover, there is a least and a greatest fixed points of f.

And the greatest is the supremum of all fixed points.

Induction vs. coinduction: lattices and fixed points

Theorem (Tarski-Knaster)

Let $\mathcal{L}$ be a complete lattice and let

$$f \ : \ \mathcal{L} \longrightarrow \mathcal{L}$$

be an order-preserving function.
Then the set of fixed points of f in $\mathcal{L}$ is also a complete lattice.

In other terms: if you define a function f on an ordered structure with supremum, infimum, least and greatest element, then it has fixed points.

Moreover, there is a least and a greatest fixed points of f.

And the greatest is the supremum of all fixed points.

Induction vs. coinduction: lattices and fixed points

Inductive constructions correspond to least fixpoints

$$\mathrm{lfp}(f) \;=\; \bigvee_i \; f^i(\bot)$$

and coinductive ones to greatest fixpoints

$$\mathrm{gfp}(f) \;=\; \bigwedge_i \; f^i(\top)$$

(note that in some cases i may have to take ordinal values. . . but not in this talk)

Induction vs. coinduction: lattices and fixed points

$$\mathsf{lfp}(f) \;=\; \bigvee_i \; f^i(\bot)$$

precisely means that inductive constructions start over some element (in a lattice $\mathcal{P}(S)$, it is the empty set), and construct iteratively a solution.

This is the spirit of the calculus of a machine.

Induction vs. coinduction: lattices and fixed points

$$\mathrm{gfp}(f) \;=\; \bigwedge_{i} f^{i}(\top)$$

precisely means that coinductive constructions start from all elements (in a lattice $\mathcal{P}(S)$, it is S), and "destruct it" iteratively until obtaining a solution.

This is the spirit of the "computation" of the configuration graph of the machine.

Induction vs. coinduction: intuitions

Two different understandings of the word infinity:

- Induction generates infinite structure, in the sense that they are unbounded.
- Coinduction generates infinite structures, in the sense that they can contain infinite (countable or more, depending on the framework) sequences.

Typically:

- Induction generates trees with arbitrary long but finite branches,
- Coinduction generates trees with countable (or even more) branches.

Induction vs. coinduction: examples

Inductive	Coinductive
Languages of words	Languages of ω-words
Finite trees	Infinite trees
Lists	Streams
Computation	Configuration graph

Other applications of coinduction

Coinduction is used to:

- Study the behavourial equivalence of (potentially infinite) processes: bisimulation relation
- Define infinite data types (infinite trees, streams. . .)
- In μ-calculus, to specify properties about infinite behaviour of programs (cf. also LTL and CTL)
- More generally, it hides in every "relation refinement" algorithm, as in the computation of the minimal automaton for example.

Centralizers are coinductive

Over the lattice $\mathcal{L} = \mathcal{P}(A^*)$, the function

$$\phi : X \mapsto (L^{-1} X) \cdot L \cap L \cdot (X L^{-1})$$

is order-preserving. As a consequence, it has fixed points, which form a complete lattice.

Notice that X is a fixed point of ϕ if and only if

$$X = (L^{-1} X) \cdot L \quad \text{and} \quad X = L \cdot (X L^{-1})$$

if and only if

$$X \cdot L = L \cdot X$$

The greatest fixpoint is $\mathcal{C}(L)$. It can be defined as the union (supremum) of all solutions.

Centralizers are coinductive

Over the lattice $\mathcal{L} = \mathcal{P}(A^*)$, the function

$$\phi \; : \; X \; \mapsto \; (L^{-1}\, X) \cdot L \cap L \cdot (X\, L^{-1})$$

is order-preserving. As a consequence, it has fixed points, which form a complete lattice.

Notice that X is a fixed point of ϕ if and only if

$$X \;\; = \;\; (L^{-1}\, X) \cdot L \quad \text{and} \quad X \;\; = \;\; L \cdot (X\, L^{-1})$$

if and only if

$$X \cdot L \;\; = \;\; L \cdot X$$

The greatest fixpoint is $\mathcal{C}(L)$. It can be defined as the union (supremum) of all solutions.

Coinduction and game interpretation

Game-theoretically, we can understand coinductive constructions as games where Eve can prove during "long-enough plays" (here, countably infinite ones) that she has a justification for her moves iff she starts from an element of the coinductive object.

Recall the situation for centralizers: starting from some word, Eve had a winning strategy iff the word was in $\mathcal{C}(L)$.

These orbits correspond to the notion of self-justifying sets, which is typical of coinduction.

Coinduction and centralizers

We can design a language L such that

1. It contains the encoding of the configurations of a circular Turing machine
2. The coinductive interpretation says that it even encodes the configuration graph of the machine
3. Two encodings are in the same commutation orbit if and only if they are in the same connected component of the configuration graph.
4. We can exclude some configurations of $\mathcal{C}(L)$

Coinduction and centralizers

Adapting Kunc's proof, we modify L so that precisely every initial configuration of the machine is removed from $\mathcal{C}(L)$.

It removes their commutation orbits: the computations of the machine.

Coinduction and centralizers

At this stage, $\mathcal{C}(L)$ contains only

- commutation orbits corresponding to infinite computations (non-terminating ones), which do not compute elements of the language of the machine,
- and commutation orbits which may reach a final configuration but not accessible from an initial configuration: that is, elements of the complementary of the machine's language.

Coinduction and centralizers

In other terms:

$\mathcal{C}(L)$ contains the encoding of the complementary of the language of a (circular) Turing machine.

Taking a universal machine gives Kunc's theorem:

$\mathcal{C}(L)$ is not recursively enumerable
(but it is co-r.e.).

Centralizers of finite languages

Recall the second part of the Theorem: L can be finite.

So far, the language we built is star-free – yet defined with stars.

It consists on a finite amount of interaction words: $f_{u,q}\, g_{u,q}$, $\widehat{d}_q$, ... used for simulating transitions, and of an infinite amount of restriction words, designed to restrict the centralizer to actual simulations of transitions.

Informally, they ensure that if you remove more than you should, then you have to remove so much that you will eventually "loose the game".

$$e_q \;\; f_q \;\; g_q \;\; a^{n+2} \;\; b \;\; \widehat{a}^{m+1}$$

Kunc gives a manner to "cut the stars" into finite words, while "forcing the players to respect them in their plays".

Centralizers of finite languages

Recall the second part of the Theorem: L can be finite.

So far, the language we built is star-free – yet defined with stars.

It consists on a finite amount of interaction words: $f_{u,q}\, g_{u,q}$, $\widehat{d}_q$, ... used for simulating transitions, and of an infinite amount of restriction words, designed to restrict the centralizer to actual simulations of transitions.

Informally, they ensure that if you remove more than you should, then you have to remove so much that you will eventually "loose the game".

$$e_q \;\; f_q \;\; g_q \;\; a^{n+2} \;\; b \;\; \widehat{a}^{m+1}$$

Kunc gives a manner to "cut the stars" into finite words, while "forcing the players to respect them in their plays".

Centralizers of finite languages

Recall the second part of the Theorem: L can be finite.

So far, the language we built is star-free – yet defined with stars.

It consists on a finite amount of interaction words: $f_{u,q}\, g_{u,q}$, $\widehat{d}_q$, ... used for simulating transitions, and of an infinite amount of restriction words, designed to restrict the centralizer to actual simulations of transitions.

Informally, they ensure that if you remove more than you should, then you have to remove so much that you will eventually "loose the game".

$$e_q \;\; f_q \;\; g_q \;\; a^{n+2} \;\; b \;\; \widehat{a}^{m+1}$$

Kunc gives a manner to "cut the stars" into finite words, while "forcing the players to respect them in their plays".

Centralizers of finite languages

Recall the second part of the Theorem: L can be finite.

So far, the language we built is star-free – yet defined with stars.

It consists on a finite amount of interaction words: $f_{u,q}\, g_{u,q}$, $\widehat{d}_q$, ... used for simulating transitions, and of an infinite amount of restriction words, designed to restrict the centralizer to actual simulations of transitions.

Informally, they ensure that if you remove more than you should, then you have to remove so much that you will eventually "loose the game".

$$e_q \; f_q \; g_q \; a^{n+2} \; b \; \widehat{a}^{m+1}$$

Kunc gives a manner to "cut the stars" into finite words, while "forcing the players to respect them in their plays".

Centralizers of finite languages

This gives a finite language L, obtained from the star-free language one. However, it requires a huge number of impossibility words.

The main reason for us to use a circular Turing machine – and not a Minsky machine – was in fact to estimate the cardinality of this finite language.

For the smallest universal Turing machine we know (4 states over a 4-symbol alphabet), it is about 10^{21} words; almost all of them are restriction words.

Centralizers of finite languages

This gives a finite language L, obtained from the star-free language one. However, it requires a huge number of impossibility words.

The main reason for us to use a circular Turing machine – and not a Minsky machine – was in fact to estimate the cardinality of this finite language.

For the smallest universal Turing machine we know (4 states over a 4-symbol alphabet), it is about 10^{21} words; almost all of them are restriction words.

Conclusion

We sketched a variant of Kunc's proof, which has three strengths:

- Only one kind of transition has to be considered, unlike for Minsky machines (or usual Turing ones)
- The coinductive nature of centralizers helps the understanding of the result and the presentation of the proof
- Cardinality of L can be estimated more accurately in the finite case.

Thank you for your attention !

Conclusion

We sketched a variant of Kunc's proof, which has three strengths:

- Only one kind of transition has to be considered, unlike for Minsky machines (or usual Turing ones)
- The coinductive nature of centralizers helps the understanding of the result and the presentation of the proof
- Cardinality of L can be estimated more accurately in the finite case.

Thank you for your attention !