# Further Remarks on Trace Languages in P Systems with Symport/Antiport

Guangwu Liu[1,2] Mihai Ionescu[1]

[1]Research Group on Mathematical Linguistics
Rovira i Virgili University
Pl. Imperial Tarraco 1, 43005 Tarragona, Spain
guangwu.liu@urv.cat, armandmihai.ionescu@urv.cat

[2]Department of Control Science and Engineering
Huazhong University of Science and Technology
Wuhan 430074, Hubei, P.R. China

Workshop on Membrane Computing 7, 2006
Leiden - The Netherlands

# Outline

# Outline

# P Systems with Symport/Antiport. Definition.

$$\Pi = (V, \mu, w_1, \ldots, w_m, E, R_1, \ldots, R_m, i_o),$$

where:

1. $V$ is the alphabet of chemicals (*objects*);
2. $\mu$ is a membrane structure with $m$ membranes; $m \geq 1$ is the *degree* of the system;
3. $w_1, \ldots, w_m$ are strings over $V$ - the multisets of objects;
4. $E$ - the *environment*;
5. $R_1, \ldots, R_m$ are finite sets of *rules* of the forms $(x, in)$, $(x, out)$, and $(x, out; y, in)$, for $x, y \in V^*$;
6. $i_o \in \{1, \ldots, m\}$ - the output membrane.

# Outline

# The idea.

- Instead of counting the number of objects
- Itineraries of a certain object through membranes[1]
- the result -> a coding of the string of labels of the visited membranes

[1] M.Ionescu, C.Martín-Vide, Gh.Păun: P systems with symport/antiport rules: The traces of objects. *Grammars*, 5, 2 (2002), 65–70.

# The idea.

- Instead of counting the number of objects
- Itineraries of a certain object through membranes[1]
- the result -> a coding of the string of labels of the visited membranes

[1] M.Ionescu, C.Martín-Vide, Gh.Păun: P systems with symport/antiport rules: The traces of objects. *Grammars*, 5, 2 (2002), 65–70.

# The idea.

- Instead of counting the number of objects
- Itineraries of a certain object through membranes[1]
- the result -> a coding of the string of labels of the visited membranes

[1] M.Ionescu, C.Martín-Vide, Gh.Păun: P systems with symport/antiport rules: The traces of objects. *Grammars*, 5, 2 (2002), 65–70.

# Definition.

$$\Pi = (V, t, T, h, \mu, w_1, \ldots, w_m, E, R_1, \ldots, R_m)$$

- $V, \mu, w_1, \ldots, w_m, E, R_1, \ldots, R_m$ are as above;
- $t \in V$ (a distinguished object,"the traveler");
- $T$ is an alphabet;
- $h : \{1, 2, \ldots, m\} \to T \cup \{\lambda\}$ is a weak coding.

The traveler is present in exactly one copy in the system - $|w_1 \ldots w_m|_t = 1$, $t \notin E$.

# Definition.

$$\Pi = (V, t, T, h, \mu, w_1, \ldots, w_m, E, R_1, \ldots, R_m)$$

- $V, \mu, w_1, \ldots, w_m, E, R_1, \ldots, R_m$ are as above;
- $t \in V$ (a distinguished object,"the traveler");
- $T$ is an alphabet;
- $h : \{1, 2, \ldots, m\} \to T \cup \{\lambda\}$ is a weak coding.

The traveler is present in exactly one copy in the system - $|w_1 \ldots w_m|_t = 1$, $t \notin E$.

# Definition.

$$\Pi = (V, t, T, h, \mu, w_1, \ldots, w_m, E, R_1, \ldots, R_m)$$

- $V, \mu, w_1, \ldots, w_m, E, R_1, \ldots, R_m$ are as above;
- $t \in V$ (a distinguished object,"the traveler");
- $T$ is an alphabet;
- $h : \{1, 2, \ldots, m\} \to T \cup \{\lambda\}$ is a weak coding.

The traveler is present in exactly one copy in the system - $|w_1 \ldots w_m|_t = 1$, $t \notin E$.

# Definition.

$$\Pi = (V, t, T, h, \mu, w_1, \ldots, w_m, E, R_1, \ldots, R_m)$$

- $V, \mu, w_1, \ldots, w_m, E, R_1, \ldots, R_m$ are as above;
- $t \in V$ (a distinguished object,"the traveler");
- $T$ is an alphabet;
- $h : \{1, 2, \ldots, m\} \to T \cup \{\lambda\}$ is a weak coding.

The traveler is present in exactly one copy in the system - $|w_1 \ldots w_m|_t = 1$, $t \notin E$.

# Definition.

$$\Pi = (V, t, T, h, \mu, w_1, \ldots, w_m, E, R_1, \ldots, R_m)$$

- $V, \mu, w_1, \ldots, w_m, E, R_1, \ldots, R_m$ are as above;
- $t \in V$ (a distinguished object,"the traveler");
- $T$ is an alphabet;
- $h : \{1, 2, \ldots, m\} \to T \cup \{\lambda\}$ is a weak coding.

The traveler is present in exactly one copy in the system - $|w_1 \ldots w_m|_t = 1$, $t \notin E$.

# The Computation.

- $\sigma = C_1 C_2 \dots C_k, k \geq 1$ - halting computation
- $C_1 = (w_1, \dots, w_m, \lambda)$ - initial configuration
- $C_i = (z_1^{(i)}, \dots, z_m^{(i)}, z_e^{(i)})$ - configuration at step $i, 1 \leq i \leq k$
  If $|z_j^{(i)}|_t = 1$ for some $1 \leq j \leq m$ we write $C_i(t) = j$
- the trace of $t$ in the computation $\sigma$:

$$trace(t, \sigma) = C_1(t) C_2(t) \dots C_k(t)$$

- computation $\sigma$ generates the string $h(trace(t, \sigma))$
- the language generated by $\Pi$ is $L(\Pi) = \{h(trace(t, \sigma)) \mid \sigma$ is a halting computation in $\Pi\}$

# The Computation.

- $\sigma = C_1 C_2 \dots C_k, k \geq 1$ - halting computation
- $C_1 = (w_1, \dots, w_m, \lambda)$ - initial configuration
- $C_i = (z_1^{(i)}, \dots, z_m^{(i)}, z_e^{(i)})$ - configuration at step $i, 1 \leq i \leq k$
  If $|z_j^{(i)}|_t = 1$ for some $1 \leq j \leq m$ we write $C_i(t) = j$
- the trace of $t$ in the computation $\sigma$:

$$trace(t, \sigma) = C_1(t) C_2(t) \dots C_k(t)$$

- computation $\sigma$ generates the string $h(trace(t, \sigma))$
- the language generated by $\Pi$ is $L(\Pi) = \{h(trace(t, \sigma)) \mid \sigma$ is a halting computation in $\Pi\}$

# The Computation.

- $\sigma = C_1 C_2 \dots C_k, k \geq 1$ - halting computation
- $C_1 = (w_1, \dots, w_m, \lambda)$ - initial configuration
- $C_i = (z_1^{(i)}, \dots, z_m^{(i)}, z_e^{(i)})$ - configuration at step $i, 1 \leq i \leq k$
  If $|z_j^{(i)}|_t = 1$ for some $1 \leq j \leq m$ we write $C_i(t) = j$
- the trace of $t$ in the computation $\sigma$:

$$trace(t, \sigma) = C_1(t) C_2(t) \dots C_k(t)$$

- computation $\sigma$ generates the string $h(trace(t, \sigma))$
- the language generated by $\Pi$ is $L(\Pi) = \{h(trace(t, \sigma)) \mid \sigma$ is a halting computation in $\Pi\}$

# The Computation.

- $\sigma = C_1 C_2 \dots C_k, k \geq 1$ - halting computation
- $C_1 = (w_1, \dots, w_m, \lambda)$ - initial configuration
- $C_i = (z_1^{(i)}, \dots, z_m^{(i)}, z_e^{(i)})$ - configuration at step $i, 1 \leq i \leq k$
  If $|z_j^{(i)}|_t = 1$ for some $1 \leq j \leq m$ we write $C_i(t) = j$
- the trace of $t$ in the computation $\sigma$:

$$trace(t, \sigma) = C_1(t) C_2(t) \dots C_k(t)$$

- computation $\sigma$ generates the string $h(trace(t, \sigma))$
- the language generated by $\Pi$ is $L(\Pi) = \{h(trace(t, \sigma)) \mid \sigma$ is a halting computation in $\Pi\}$

# The Computation.

- $\sigma = C_1 C_2 \dots C_k, k \geq 1$ - halting computation
- $C_1 = (w_1, \dots, w_m, \lambda)$ - initial configuration
- $C_i = (z_1^{(i)}, \dots, z_m^{(i)}, z_e^{(i)})$ - configuration at step $i, 1 \leq i \leq k$
  If $|z_j^{(i)}|_t = 1$ for some $1 \leq j \leq m$ we write $C_i(t) = j$
- the trace of $t$ in the computation $\sigma$:

$$trace(t, \sigma) = C_1(t) C_2(t) \dots C_k(t)$$

- computation $\sigma$ generates the string $h(trace(t, \sigma))$
- the language generated by $\Pi$ is $L(\Pi) = \{h(trace(t, \sigma)) \mid \sigma$ is a halting computation in $\Pi\}$

# The Computation.

- $\sigma = C_1 C_2 \dots C_k, k \geq 1$ - halting computation
- $C_1 = (w_1, \dots, w_m, \lambda)$ - initial configuration
- $C_i = (z_1^{(i)}, \dots, z_m^{(i)}, z_e^{(i)})$ - configuration at step $i, 1 \leq i \leq k$
  If $|z_j^{(i)}|_t = 1$ for some $1 \leq j \leq m$ we write $C_i(t) = j$
- the trace of $t$ in the computation $\sigma$:

$$trace(t, \sigma) = C_1(t) C_2(t) \dots C_k(t)$$

- computation $\sigma$ generates the string $h(trace(t, \sigma))$
- the language generated by $\Pi$ is $L(\Pi) = \{h(trace(t, \sigma)) \mid \sigma$ is a halting computation in $\Pi\}$

# Currently Best Results.

- $RE = LTP_*(sym_0, anti_2)^2$
- $IRE = ILTP_{l+1}(sym_0, anti_2)$
- $IRE = ILTP_{l+1}(sym_3, anti_0)$
- $IRE = ILTP_{l+2}(sym_2, anti_0)$

[2] P. Frisco, H.J. Hoogeboom: P systems with symport/antiport simulating counter automata, *Acta Informatica*, 41(2-3), 2004, 145–170.

# Currently Best Results.

- $RE = LTP_*(sym_0, anti_2)^2$
- $IRE = ILTP_{l+1}(sym_0, anti_2)$
- $IRE = ILTP_{l+1}(sym_3, anti_0)$
- $IRE = ILTP_{l+2}(sym_2, anti_0)$

[2] P. Frisco, H.J. Hoogeboom: P systems with symport/antiport simulating counter automata, *Acta Informatica*, 41(2-3), 2004, 145–170.

# Currently Best Results.

- $RE = LTP_*(sym_0, anti_2)^2$
- $IRE = ILTP_{l+1}(sym_0, anti_2)$
- $IRE = ILTP_{l+1}(sym_3, anti_0)$
- $IRE = ILTP_{l+2}(sym_2, anti_0)$

[2] P. Frisco, H.J. Hoogeboom: P systems with symport/antiport simulating counter automata, *Acta Informatica*, 41(2-3), 2004, 145–170.

# Currently Best Results.

- $RE = LTP_*(sym_0, anti_2)^2$
- $IRE = ILTP_{l+1}(sym_0, anti_2)$
- $IRE = ILTP_{l+1}(sym_3, anti_0)$
- $IRE = ILTP_{l+2}(sym_2, anti_0)$

[2] P. Frisco, H.J. Hoogeboom: P systems with symport/antiport simulating counter automata, *Acta Informatica*, 41(2-3), 2004, 145–170.

# Outline

# Definition I.

$$\Pi = (V, t, \mu, w_1, \ldots, w_m, R_1, \ldots, R_m),$$

1. $V$ – alphabet of chemicals (objects);
2. $t \in V$ – the traveler; $|w_1 \cdots w_m|_t = 1$;
3. $\mu$ – membrane structure with $m$ membranes;
4. $w_i$ – strings representing the sets of objects present in the regions of $\mu$, $1 \leq i \leq m$;
5. $R_i$ – set of symport and antiport rules for membrane $i$; $(x, in)$, $(x, out)$ and $(x, out; y, in)$, for $x, y \in V^*$, $1 \leq i \leq m$.

The trace: encoded as a string over $\{a_1, a_2, \ldots, a_m\}$ marking every membrane visited by $t$.

# Definition I.

$$\Pi = (V, t, \mu, w_1, \ldots, w_m, R_1, \ldots, R_m),$$

1. $V$ – alphabet of chemicals (objects);
2. $t \in V$ – the traveler; $|w_1 \cdots w_m|_t = 1$;
3. $\mu$ – membrane structure with $m$ membranes;
4. $w_i$ – strings representing the sets of objects present in the regions of $\mu$, $1 \leq i \leq m$;
5. $R_i$ – set of symport and antiport rules for membrane $i$; $(x, in)$, $(x, out)$ and $(x, out; y, in)$, for $x, y \in V^*$, $1 \leq i \leq m$.

The trace: encoded as a string over $\{a_1, a_2, \ldots, a_m\}$ marking every membrane visited by $t$.

# Definition I.

$$\Pi = (V, t, \mu, w_1, \ldots, w_m, R_1, \ldots, R_m),$$

1. $V$ – alphabet of chemicals (objects);
2. $t \in V$ – the traveler; $|w_1 \cdots w_m|_t = 1$;
3. $\mu$ – membrane structure with $m$ membranes;
4. $w_i$ – strings representing the sets of objects present in the regions of $\mu$, $1 \leq i \leq m$;
5. $R_i$ – set of symport and antiport rules for membrane $i$; $(x, in)$, $(x, out)$ and $(x, out; y, in)$, for $x, y \in V^*$, $1 \leq i \leq m$.

The trace: encoded as a string over $\{a_1, a_2, \ldots, a_m\}$ marking every membrane visited by $t$.

# Definition I.

$$\Pi = (V, t, \mu, w_1, \ldots, w_m, R_1, \ldots, R_m),$$

1. $V$ – alphabet of chemicals (objects);
2. $t \in V$ – the traveler; $|w_1 \cdots w_m|_t = 1$;
3. $\mu$ – membrane structure with $m$ membranes;
4. $w_i$ – strings representing the sets of objects present in the regions of $\mu$, $1 \leq i \leq m$;
5. $R_i$ – set of symport and antiport rules for membrane $i$; $(x, in), (x, out)$ and $(x, out; y, in)$, for $x, y \in V^*$, $1 \leq i \leq m$.

The trace: encoded as a string over $\{a_1, a_2, \ldots, a_m\}$ marking every membrane visited by $t$.

# Definition I.

$$\Pi = (V, t, \mu, w_1, \ldots, w_m, R_1, \ldots, R_m),$$

1. $V$ – alphabet of chemicals (objects);
2. $t \in V$ – the traveler; $|w_1 \cdots w_m|_t = 1$;
3. $\mu$ – membrane structure with $m$ membranes;
4. $w_i$ – strings representing the sets of objects present in the regions of $\mu$, $1 \leq i \leq m$;
5. $R_i$ – set of symport and antiport rules for membrane $i$; $(x, in), (x, out)$ and $(x, out; y, in)$, for $x, y \in V^*$, $1 \leq i \leq m$.

The trace: encoded as a string over $\{a_1, a_2, \ldots, a_m\}$ marking every membrane visited by $t$.

# Definition II.

Two cases of marking the trace:

- only when the traveler enters a membrane:

$$LTP_m^{set}(sym_p, anti_q, in)$$

- both when the traveler enters and exits a membrane (we collect twice the label of the membrane $t$ visits):

$$LTP_m^{set}(sym_p, anti_q, in/out)$$

# Definition II.

Two cases of marking the trace:

- only when the traveler enters a membrane:

$$LTP_m^{set}(sym_p, anti_q, in)$$

- both when the traveler enters and exits a membrane (we collect twice the label of the membrane $t$ visits):

$$LTP_m^{set}(sym_p, anti_q, in/out)$$

# Outline

# Results I.

## Theorem

$LTP_*^{set}(sym_*, anti_*, in) = REG.$

$$A \to a_i B \in R$$

| step | $R_0$ | $R_i$ |
|---|---|---|
| 1 | | $(f_i, out; cAt, in)$ |
| 2 | $(d, out; f_i, in)$ | $(At, out)$ |
| 3 | $(f_i B', out; A, in)$ | $(c, out; d, in)$ |
| 4 | | $(d, out; f_i, in)$ |
| 5 | $(B, out; dB', in)$ | |

degree of the systems and the weight of rules:

$$mREG = LTP_{m+2}^{set}(sym_2, anti_3, in)$$

# Results II.

**Theorem**

$LTP_*^{set}(sym_*, anti_*, in/out) \subsetneq REG$.

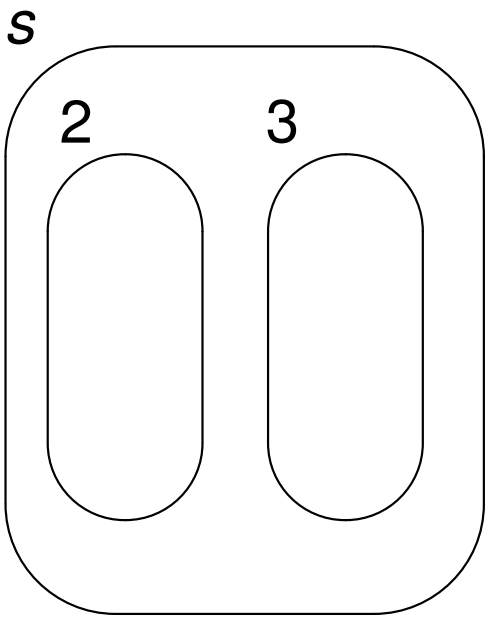
Strict inclusion. $L = \{a_2a_3, a_2a_2a_3\}$

# Results II.

**Theorem**

$LTP_*^{set}(sym_*, anti_*, in/out) \subsetneq REG$.

Strict inclusion. $L = \{a_2a_3, a_2a_2a_3\}$

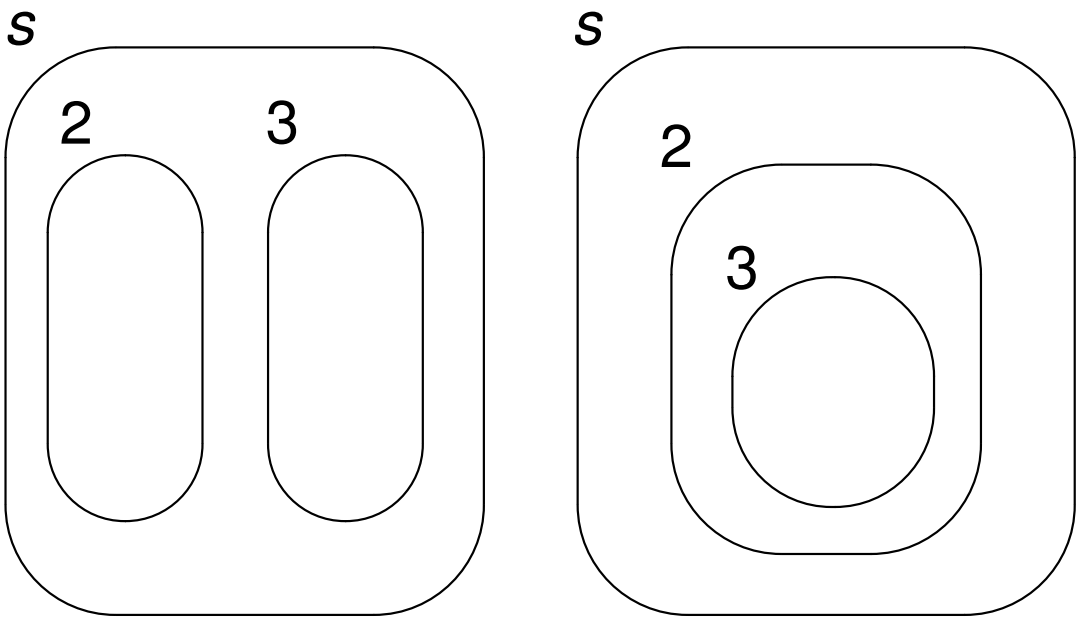


# Results II.

**Theorem**

$LTP_*^{set}(sym_*, anti_*, in/out) \subsetneq REG$.

Strict inclusion. $L = \{a_2a_3, a_2a_2a_3\}$

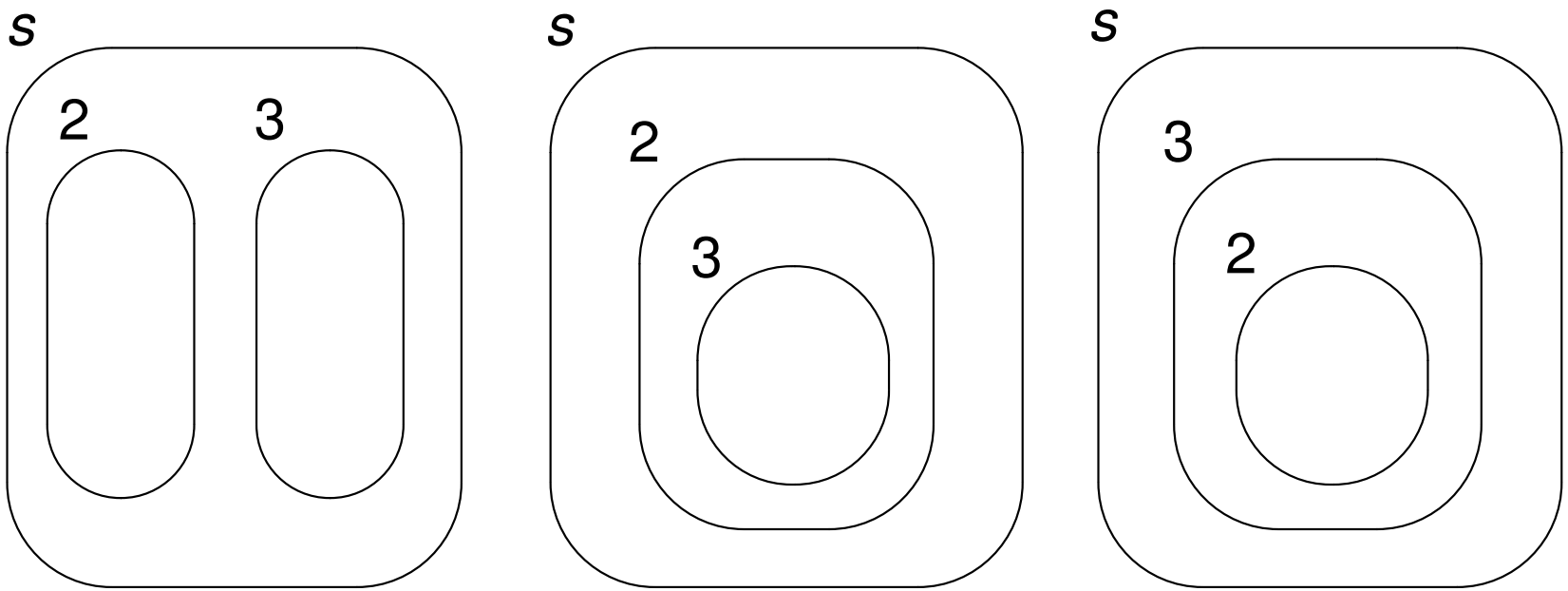


# Results II.

**Theorem**

$LTP_*^{set}(sym_*, anti_*, in/out) \subsetneq REG$.

Strict inclusion. $L = \{a_2a_3, a_2a_2a_3\}$

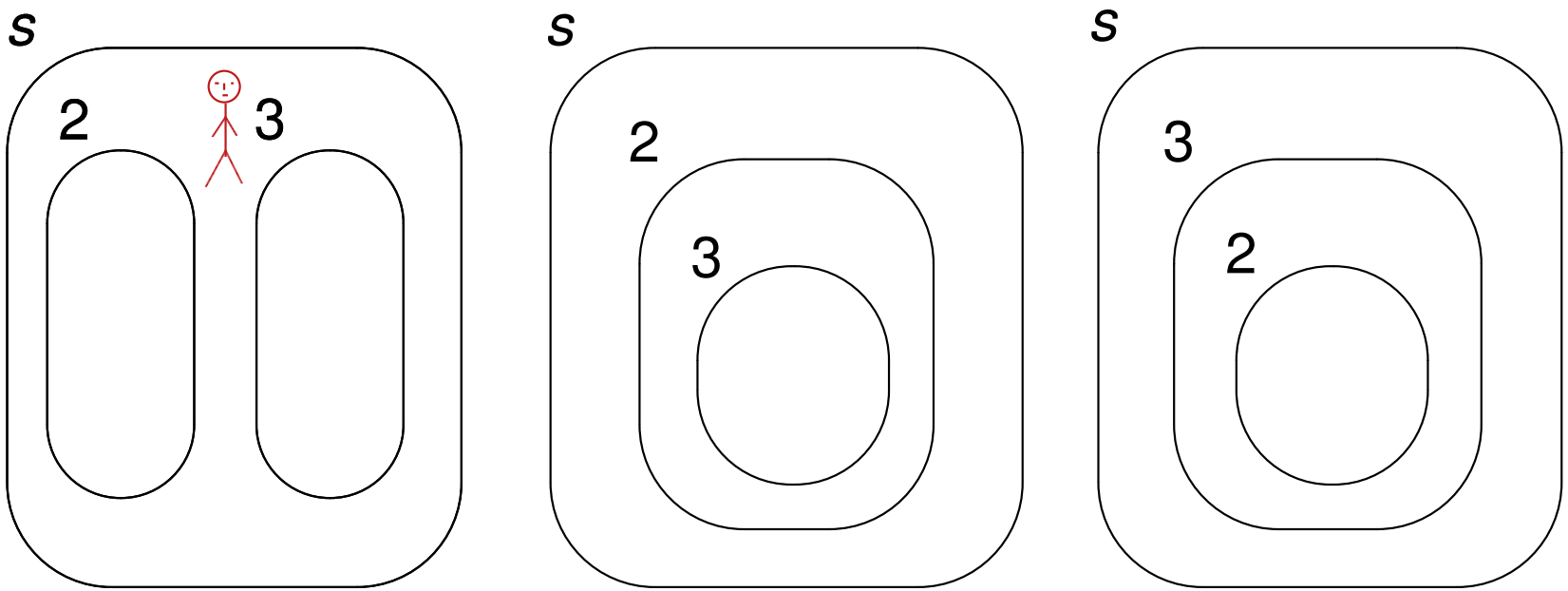


# Results II.

**Theorem**

$LTP_*^{set}(sym_*, anti_*, in/out) \subsetneq REG$.

Strict inclusion. $L = \{a_2a_3, a_2a_2a_3\}$

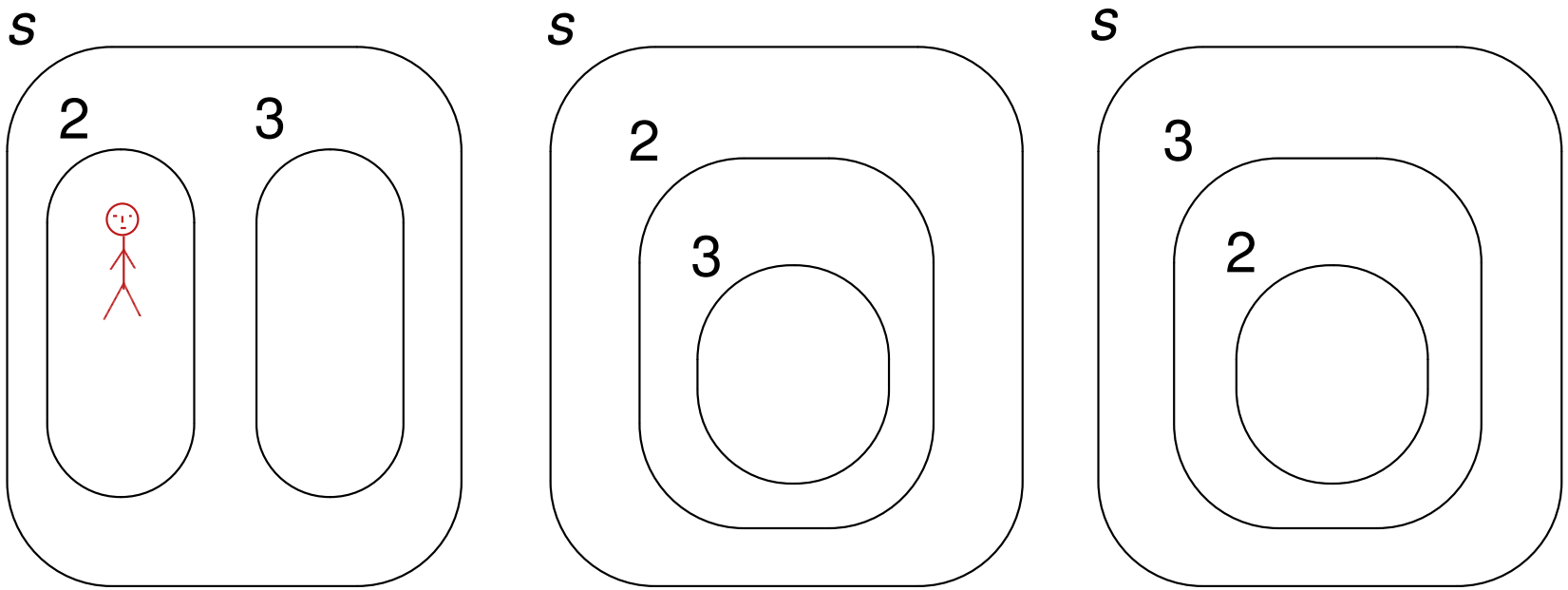


# Results II.

**Theorem**

$LTP_*^{set}(sym_*, anti_*, in/out) \subsetneq REG$.

Strict inclusion. $L = \{a_2a_3, a_2a_2a_3\}$ $a_2$

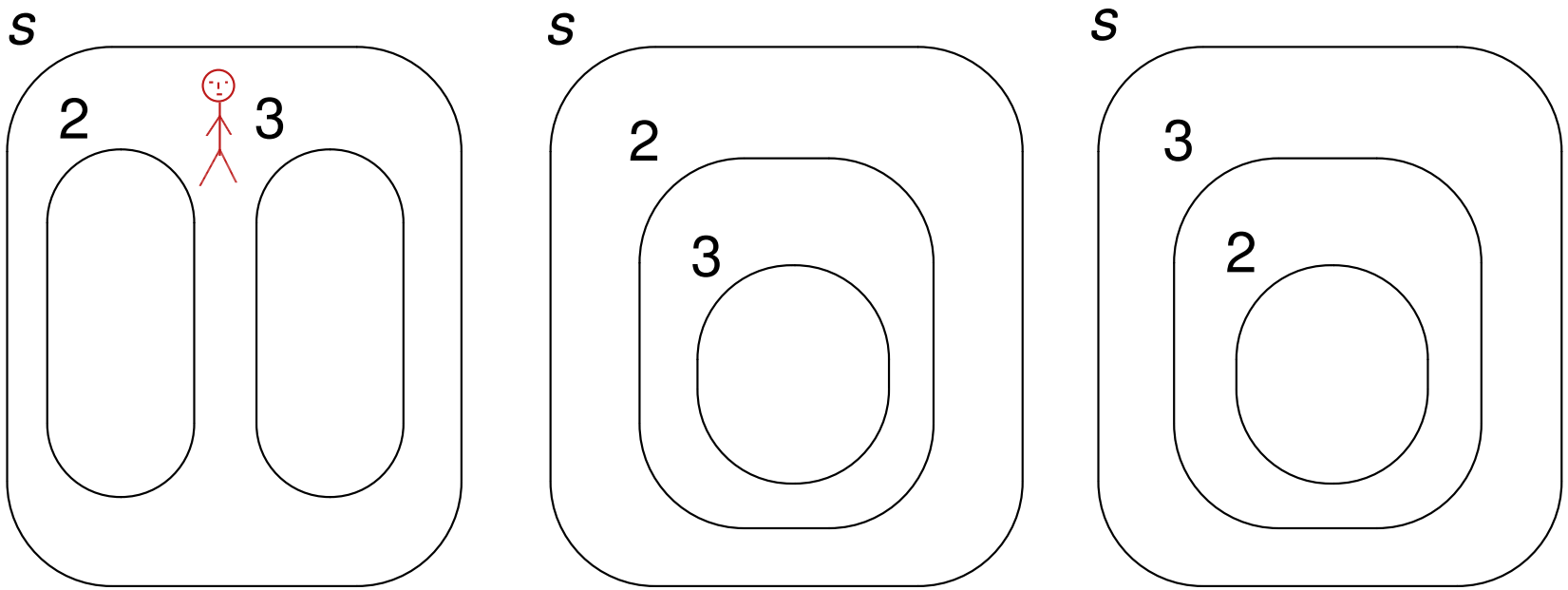


# Results II.

**Theorem**

$LTP_*^{set}(sym_*, anti_*, in/out) \subsetneq REG$.

Strict inclusion. $L = \{a_2a_3, a_2a_2a_3\}$ $a_2a_2$

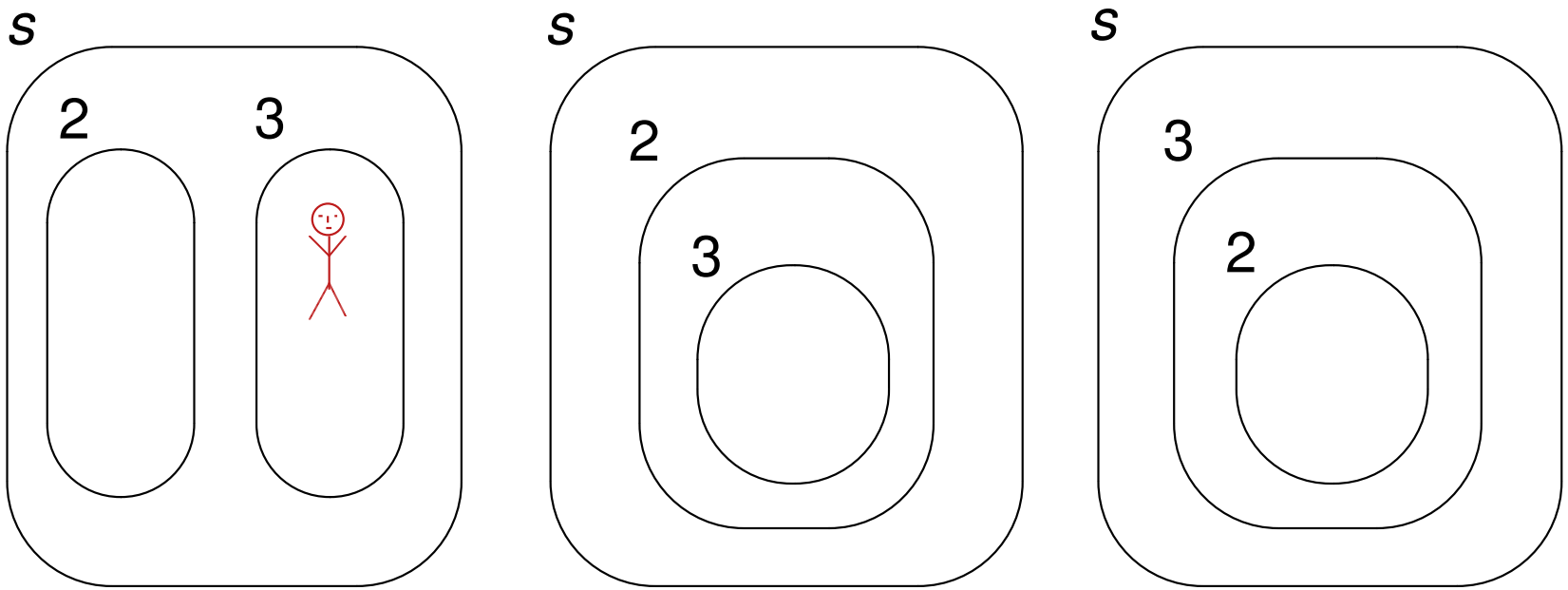


# Results II.

**Theorem**

$LTP_*^{set}(sym_*, anti_*, in/out) \subsetneq REG$.

Strict inclusion. $L = \{a_2a_3, a_2a_2a_3\}$ $a_2a_2a_3$



# Outline

# Idea.

- $T = \{t_1, t_2, \ldots, t_k\}$ - $k$ travelers
- $1, 2, \ldots, m$ - membranes
- $\Rightarrow$ alphabet of trace symbols: $k \cdot m$ symbols $a_{i,j}$, $1 \leq i \leq k, 1 \leq j \leq m$
- $\sigma = C_1 C_2 \ldots C_k,\ k \geq 1$ halting computation
- $C_i(T) = \{a_{i,j} \mid t_i$ is in membrane $j\}$
- $trace(T, \sigma) = \{w_1 w_2 \ldots w_k \mid w_i \in V^*, \Psi_V(w_i) = \Psi_V(C_i(T))\}$, i.e concatenation of permutations of strings
- trace language: $LT(\Pi) = \{h(trace(T, \sigma)) \mid \sigma$ is a halting computation in $\Pi\}$

Remark: $k$ travelers, $m$ membranes $\Rightarrow LT(\Pi)$ can be on $k \cdot m$ symbols

# Idea.

- $T = \{t_1, t_2, \ldots, t_k\}$ - $k$ travelers
- $1, 2, \ldots, m$ - membranes
- $\Rightarrow$ alphabet of trace symbols: $k \cdot m$ symbols $a_{i,j}$, $1 \leq i \leq k, 1 \leq j \leq m$
- $\sigma = C_1 C_2 \ldots C_k,\ k \geq 1$ halting computation
- $C_i(T) = \{a_{i,j} \mid t_i$ is in membrane $j\}$
- $trace(T, \sigma) = \{w_1 w_2 \ldots w_k \mid w_i \in V^*, \Psi_V(w_i) = \Psi_V(C_i(T))\}$, i.e concatenation of permutations of strings
- trace language: $LT(\Pi) = \{h(trace(T, \sigma)) \mid \sigma$ is a halting computation in $\Pi\}$

Remark: $k$ travelers, $m$ membranes $\Rightarrow LT(\Pi)$ can be on $k \cdot m$ symbols

# Idea.

- $T = \{t_1, t_2, \ldots, t_k\}$ - $k$ travelers
- $1, 2, \ldots, m$ - membranes
- $\Rightarrow$ alphabet of trace symbols: $k \cdot m$ symbols $a_{i,j}$, $1 \leq i \leq k$, $1 \leq j \leq m$
- $\sigma = C_1 C_2 \ldots C_k$, $k \geq 1$ halting computation
- $C_i(T) = \{a_{i,j} \mid t_i$ is in membrane $j\}$
- $trace(T, \sigma) = \{w_1 w_2 \ldots w_k \mid w_i \in V^*, \Psi_V(w_i) = \Psi_V(C_i(T))\}$, i.e concatenation of permutations of strings
- trace language: $LT(\Pi) = \{h(trace(T, \sigma)) \mid \sigma$ is a halting computation in $\Pi\}$

Remark: $k$ travelers, $m$ membranes $\Rightarrow LT(\Pi)$ can be on $k \cdot m$ symbols

# Idea.

- $T = \{t_1, t_2, \ldots, t_k\}$ - $k$ travelers
- $1, 2, \ldots, m$ - membranes
- $\Rightarrow$ alphabet of trace symbols: $k \cdot m$ symbols $a_{i,j}$, $1 \leq i \leq k, 1 \leq j \leq m$
- $\sigma = C_1 C_2 \ldots C_k,\ k \geq 1$ halting computation
- $C_i(T) = \{a_{i,j} \mid t_i$ is in membrane $j\}$
- $trace(T, \sigma) = \{w_1 w_2 \ldots w_k \mid w_i \in V^*, \Psi_V(w_i) = \Psi_V(C_i(T))\}$, i.e concatenation of permutations of strings
- trace language: $LT(\Pi) = \{h(trace(T, \sigma)) \mid \sigma$ is a halting computation in $\Pi\}$

Remark: $k$ travelers, $m$ membranes $\Rightarrow LT(\Pi)$ can be on $k \cdot m$ symbols

# Idea.

- $T = \{t_1, t_2, \ldots, t_k\}$ - $k$ travelers
- $1, 2, \ldots, m$ - membranes
- $\Rightarrow$ alphabet of trace symbols: $k \cdot m$ symbols $a_{i,j}$, $1 \leq i \leq k$, $1 \leq j \leq m$
- $\sigma = C_1 C_2 \ldots C_k,\ k \geq 1$ halting computation
- $C_i(T) = \{a_{i,j} \mid t_i$ is in membrane $j\}$
- $trace(T, \sigma) = \{w_1 w_2 \ldots w_k \mid w_i \in V^*, \Psi_V(w_i) = \Psi_V(C_i(T))\}$, i.e concatenation of permutations of strings
- trace language: $LT(\Pi) = \{h(trace(T, \sigma)) \mid \sigma$ is a halting computation in $\Pi\}$

Remark: $k$ travelers, $m$ membranes $\Rightarrow LT(\Pi)$ can be on $k \cdot m$ symbols

# Idea.

- $T = \{t_1, t_2, \ldots, t_k\}$ - $k$ travelers
- $1, 2, \ldots, m$ - membranes
- $\Rightarrow$ alphabet of trace symbols: $k \cdot m$ symbols $a_{i,j}$, $1 \leq i \leq k$, $1 \leq j \leq m$
- $\sigma = C_1 C_2 \ldots C_k,\ k \geq 1$ halting computation
- $C_i(T) = \{a_{i,j} \mid t_i$ is in membrane $j\}$
- $trace(T, \sigma) = \{w_1 w_2 \ldots w_k \mid w_i \in V^*, \Psi_V(w_i) = \Psi_V(C_i(T))\}$, i.e concatenation of permutations of strings
- trace language: $LT(\Pi) = \{h(trace(T, \sigma)) \mid \sigma$ is a halting computation in $\Pi\}$

Remark: $k$ travelers, $m$ membranes $\Rightarrow LT(\Pi)$ can be on $k \cdot m$ symbols

# Idea.

- $T = \{t_1, t_2, \ldots, t_k\}$ - $k$ travelers
- $1, 2, \ldots, m$ - membranes
- $\Rightarrow$ alphabet of trace symbols: $k \cdot m$ symbols $a_{i,j}$, $1 \leq i \leq k$, $1 \leq j \leq m$
- $\sigma = C_1 C_2 \ldots C_k,\ k \geq 1$ halting computation
- $C_i(T) = \{a_{i,j} \mid t_i$ is in membrane $j\}$
- $trace(T, \sigma) = \{w_1 w_2 \ldots w_k \mid w_i \in V^*, \Psi_V(w_i) = \Psi_V(C_i(T))\}$, i.e concatenation of permutations of strings
- trace language: $LT(\Pi) = \{h(trace(T, \sigma)) \mid \sigma$ is a halting computation in $\Pi\}$

Remark: $k$ travelers, $m$ membranes $\Rightarrow LT(\Pi)$ can be on $k \cdot m$ symbols

# Idea.

- $T = \{t_1, t_2, \ldots, t_k\}$ - $k$ travelers
- $1, 2, \ldots, m$ - membranes
- $\Rightarrow$ alphabet of trace symbols: $k \cdot m$ symbols $a_{i,j}$, $1 \leq i \leq k$, $1 \leq j \leq m$
- $\sigma = C_1 C_2 \ldots C_k$, $k \geq 1$ halting computation
- $C_i(T) = \{a_{i,j} \mid t_i$ is in membrane $j\}$
- $trace(T, \sigma) = \{w_1 w_2 \ldots w_k \mid w_i \in V^*, \Psi_V(w_i) = \Psi_V(C_i(T))\}$, i.e concatenation of permutations of strings
- trace language: $LT(\Pi) = \{h(trace(T, \sigma)) \mid \sigma$ is a halting computation in $\Pi\}$

Remark: $k$ travelers, $m$ membranes $\Rightarrow LT(\Pi)$ can be on $k \cdot m$ symbols

# Outline

# Idea.

- $h : V^* \to U^*$,
  $h^{-1} : U^* \to 2^{V^*}; h^{-1}(y) = \{x \in V^* \mid h(x) = y\}, y \in U^*$

**Example**

- $V = \{a_1, a_2, \ldots, a_m\}$
- $U = \{0, 1\}$
- $h(a_i) = 0^i 1, 1 \leq i \leq m$ !Injective, hence $card(h^{-1}(y)) = 1$ for each $y \in h(V^*)$
- $\Rightarrow$ for any language $L \subseteq V^*$ -> $L = h^{-1}(h(L))$
- $L \in mFL \Rightarrow h(L) \in 2FL$
- because $mREG = LTP^{set}_{m+2}(sym_2, anti_3, in) \Rightarrow$
  $h(L) \in 2LTP^{set}_4(sym_2, anti_3, in)$

**Proposition:** Every $L \in mREG$ can be written in the form $L = h^{-1}(L')$, for $L' \in 2LTP^{set}_4(sym_2, anti_3, in)$.

# Idea.

- $h : V^* \to U^*$,
  $h^{-1} : U^* \to 2^{V^*}; h^{-1}(y) = \{x \in V^* \mid h(x) = y\}, y \in U^*$

## Example

- $V = \{a_1, a_2, \ldots, a_m\}$
- $U = \{0, 1\}$
- $h(a_i) = 0^i 1, 1 \leq i \leq m$ !Injective, hence $card(h^{-1}(y)) = 1$ for each $y \in h(V^*)$
- $\Rightarrow$ for any language $L \subseteq V^*$ -> $L = h^{-1}(h(L))$
- $L \in mFL \Rightarrow h(L) \in 2FL$
- because $mREG = LTP^{set}_{m+2}(sym_2, anti_3, in) \Rightarrow$
  $h(L) \in 2LTP^{set}_4(sym_2, anti_3, in)$

Proposition: Every $L \in mREG$ can be written in the form $L = h^{-1}(L')$, for $L' \in 2LTP^{set}_4(sym_2, anti_3, in)$.

# Idea.

- $h : V^* \to U^*$,
  $h^{-1} : U^* \to 2^{V^*}; h^{-1}(y) = \{x \in V^* \mid h(x) = y\}, y \in U^*$

## Example

- $V = \{a_1, a_2, \ldots, a_m\}$
- $U = \{0, 1\}$
- $h(a_i) = 0^i 1, 1 \leq i \leq m$ !Injective, hence $card(h^{-1}(y)) = 1$ for each $y \in h(V^*)$
- $\Rightarrow$ for any language $L \subseteq V^*$ -> $L = h^{-1}(h(L))$
- $L \in mFL \Rightarrow h(L) \in 2FL$
- because $mREG = LTP^{set}_{m+2}(sym_2, anti_3, in) \Rightarrow$
  $h(L) \in 2LTP^{set}_4(sym_2, anti_3, in)$

Proposition: Every $L \in mREG$ can be written in the form $L = h^{-1}(L')$, for $L' \in 2LTP^{set}_4(sym_2, anti_3, in)$.

# Idea.

- $h : V^* \to U^*$,
  $h^{-1} : U^* \to 2^{V^*}; h^{-1}(y) = \{x \in V^* \mid h(x) = y\}, y \in U^*$

## Example

- $V = \{a_1, a_2, \ldots, a_m\}$
- $U = \{0, 1\}$
- $h(a_i) = 0^i 1, 1 \leq i \leq m$ !Injective, hence $card(h^{-1}(y)) = 1$ for each $y \in h(V^*)$
- $\Rightarrow$ for any language $L \subseteq V^*$ -> $L = h^{-1}(h(L))$
- $L \in mFL \Rightarrow h(L) \in 2FL$
- because $mREG = LTP^{set}_{m+2}(sym_2, anti_3, in) \Rightarrow$ $h(L) \in 2LTP^{set}_4(sym_2, anti_3, in)$

Proposition: Every $L \in mREG$ can be written in the form $L = h^{-1}(L')$, for $L' \in 2LTP^{set}_4(sym_2, anti_3, in)$.

# Idea.

- $h : V^* \to U^*$,
  $h^{-1} : U^* \to 2^{V^*}; h^{-1}(y) = \{x \in V^* \mid h(x) = y\}, y \in U^*$

## Example

- $V = \{a_1, a_2, \ldots, a_m\}$
- $U = \{0, 1\}$
- $h(a_i) = 0^i 1, 1 \leq i \leq m$ !Injective, hence $card(h^{-1}(y)) = 1$ for each $y \in h(V^*)$
- $\Rightarrow$ for any language $L \subseteq V^*$ -> $L = h^{-1}(h(L))$
- $L \in mFL \Rightarrow h(L) \in 2FL$
- because $mREG = LTP^{set}_{m+2}(sym_2, anti_3, in) \Rightarrow$
  $h(L) \in 2LTP^{set}_4(sym_2, anti_3, in)$

Proposition: Every $L \in mREG$ can be written in the form $L = h^{-1}(L')$, for $L' \in 2LTP^{set}_4(sym_2, anti_3, in)$.

# Idea.

- $h : V^* \to U^*$,
  $h^{-1} : U^* \to 2^{V^*}; h^{-1}(y) = \{x \in V^* \mid h(x) = y\}, y \in U^*$

## Example

- $V = \{a_1, a_2, \ldots, a_m\}$
- $U = \{0, 1\}$
- $h(a_i) = 0^i 1, 1 \leq i \leq m$ !Injective, hence $card(h^{-1}(y)) = 1$ for each $y \in h(V^*)$
- $\Rightarrow$ for any language $L \subseteq V^*$ -> $L = h^{-1}(h(L))$
- $L \in mFL \Rightarrow h(L) \in 2FL$
- because $mREG = LTP^{set}_{m+2}(sym_2, anti_3, in) \Rightarrow h(L) \in 2LTP^{set}_4(sym_2, anti_3, in)$

Proposition: Every $L \in mREG$ can be written in the form $L = h^{-1}(L')$, for $L' \in 2LTP^{set}_4(sym_2, anti_3, in)$.

# Idea.

- $h : V^* \to U^*$,
  $h^{-1} : U^* \to 2^{V^*}; h^{-1}(y) = \{x \in V^* \mid h(x) = y\}, y \in U^*$

## Example

- $V = \{a_1, a_2, \ldots, a_m\}$
- $U = \{0, 1\}$
- $h(a_i) = 0^i 1, 1 \leq i \leq m$ !Injective, hence $card(h^{-1}(y)) = 1$ for each $y \in h(V^*)$
- $\Rightarrow$ for any language $L \subseteq V^*$ -> $L = h^{-1}(h(L))$
- $L \in mFL \Rightarrow h(L) \in 2FL$
- because $mREG = LTP^{set}_{m+2}(sym_2, anti_3, in) \Rightarrow$
  $h(L) \in 2LTP^{set}_4(sym_2, anti_3, in)$

Proposition: Every $L \in mREG$ can be written in the form $L = h^{-1}(L')$, for $L' \in 2LTP^{set}_4(sym_2, anti_3, in)$.

# Idea.

- $h : V^* \to U^*$,
  $h^{-1} : U^* \to 2^{V^*}; h^{-1}(y) = \{x \in V^* \mid h(x) = y\}, y \in U^*$

## Example

- $V = \{a_1, a_2, \ldots, a_m\}$
- $U = \{0, 1\}$
- $h(a_i) = 0^i 1, 1 \leq i \leq m$ !Injective, hence $card(h^{-1}(y)) = 1$ for each $y \in h(V^*)$
- $\Rightarrow$ for any language $L \subseteq V^*$ -> $L = h^{-1}(h(L))$
- $L \in mFL \Rightarrow h(L) \in 2FL$
- because $mREG = LTP^{set}_{m+2}(sym_2, anti_3, in) \Rightarrow$
  $h(L) \in 2LTP^{set}_4(sym_2, anti_3, in)$

**Proposition:** Every $L \in mREG$ can be written in the form $L = h^{-1}(L')$, for $L' \in 2LTP^{set}_4(sym_2, anti_3, in)$.

# Outline

# Idea.

- Rewrite:
  $(x, in)$: $x[\ ]_i \to [x]_i$
  $(x, out)$: $[x]_i \to [\ ]_i x$
  $(x, out; y, in)$: $y[x]_i \to [y]_i x$
- Generalize:
  $x[\ ]_i \to [x]_j$
  $[x]_i \to [\ ]_j x$
  $y[x]_i \to [y]_j x$
- $\Rightarrow$ conflict with the new labels
- Solutions to avoid conflict:
  - sequential use of the rules
  - use only $(i, j)$-coherent set of rules, all passing from $i$ to $j$
  - allow only to certain rules to change the labels

# Idea.

- Rewrite:
  $(x, in)$: $x[\ ]_i \to [x]_i$
  $(x, out)$: $[x]_i \to [\ ]_i x$
  $(x, out; y, in)$: $y[x]_i \to [y]_i x$
- Generalize:
  $x[\ ]_i \to [x]_j$
  $[x]_i \to [\ ]_j x$
  $y[x]_i \to [y]_j x$
- $\Rightarrow$ conflict with the new labels
- Solutions to avoid conflict:
  - sequential use of the rules
  - use only $(i, j)$-coherent set of rules, all passing from $i$ to $j$
  - allow only to certain rules to change the labels

# Idea.

- Rewrite:
  $(x, in)$: $x[\ ]_i \to [x]_i$
  $(x, out)$: $[x]_i \to [\ ]_i x$
  $(x, out; y, in)$: $y[x]_i \to [y]_i x$
- Generalize:
  $x[\ ]_i \to [x]_j$
  $[x]_i \to [\ ]_j x$
  $y[x]_i \to [y]_j x$
- $\Rightarrow$ conflict with the new labels
- Solutions to avoid conflict:
  - sequential use of the rules
  - use only $(i, j)$-coherent set of rules, all passing from $i$ to $j$
  - allow only to certain rules to change the labels

# Idea.

- Rewrite:
  $(x, in)$: $x[\ ]_i \to [x]_i$
  $(x, out)$: $[x]_i \to [\ ]_i x$
  $(x, out; y, in)$: $y[x]_i \to [y]_i x$
- Generalize:
  $x[\ ]_i \to [x]_j$
  $[x]_i \to [\ ]_j x$
  $y[x]_i \to [y]_j x$
- $\Rightarrow$ conflict with the new labels
- Solutions to avoid conflict:
  - sequential use of the rules
  - use only $(i, j)$-coherent set of rules, all passing from $i$ to $j$
  - allow only to certain rules to change the labels

# Idea.

- Rewrite:
  $(x, in)$: $x[\ ]_i \to [x]_i$
  $(x, out)$: $[x]_i \to [\ ]_i x$
  $(x, out; y, in)$: $y[x]_i \to [y]_i x$
- Generalize:
  $x[\ ]_i \to [x]_j$
  $[x]_i \to [\ ]_j x$
  $y[x]_i \to [y]_j x$
- $\Rightarrow$ conflict with the new labels
- Solutions to avoid conflict:
  - sequential use of the rules
  - use only $(i, j)$-coherent set of rules, all passing from $i$ to $j$
  - allow only to certain rules to change the labels

# Idea.

- Rewrite:
  $(x, in)$: $x[\ ]_i \to [x]_i$
  $(x, out)$: $[x]_i \to [\ ]_i x$
  $(x, out; y, in)$: $y[x]_i \to [y]_i x$
- Generalize:
  $x[\ ]_i \to [x]_j$
  $[x]_i \to [\ ]_j x$
  $y[x]_i \to [y]_j x$
- $\Rightarrow$ conflict with the new labels
- Solutions to avoid conflict:
  - sequential use of the rules
  - use only $(i, j)$-coherent set of rules, all passing from $i$ to $j$
  - allow only to certain rules to change the labels

# Idea.

- Rewrite:
  $(x, in)$: $x[\ ]_i \to [x]_i$
  $(x, out)$: $[x]_i \to [\ ]_i x$
  $(x, out; y, in)$: $y[x]_i \to [y]_i x$
- Generalize:
  $x[\ ]_i \to [x]_j$
  $[x]_i \to [\ ]_j x$
  $y[x]_i \to [y]_j x$
- $\Rightarrow$ conflict with the new labels
- Solutions to avoid conflict:
  - sequential use of the rules
  - use only $(i, j)$-coherent set of rules, all passing from $i$ to $j$
  - allow only to certain rules to change the labels

# Conclusions

- P systems with sets of objects. Computational power does not go beyond *REG*.
- Three ideas to break the infinite hierarchy provoked by cardinality of the alphabet of languages - no. of membranes.

# The end.

THANK YOU!