

NETWORKS OF CELL-LIKE PROCESSORS

Victor Mitrana

Faculty of Mathematics and Computer Science

University of Bucharest, Romania

mitrana@fmi.unibuc.ro

GENERAL IDEA

A set of nodes that are connected: a network

Each node: evolutionary/splicing processor

Two distinguished nodes: In and Out

The edges: bidirectional communication channels

Information: strings

Restricted communication by: input/output filters.

Input: a string in *In*.

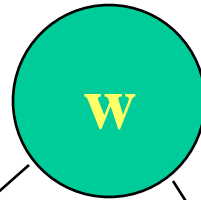
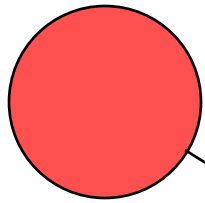
Computation: Evolution/Splicing, Communication,

Halting: a string enters *Out* or two consecutive identical configurations.

Acceptance: a string enters *Out*

ANEP

Output node

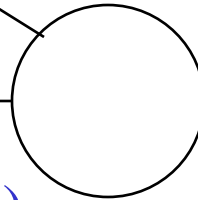
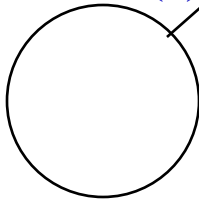


Input node

Substitution: $a \rightarrow b$
 $a \rightarrow c$

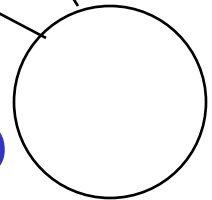
Deletion: $a \rightarrow \lambda (*)$
 $b \rightarrow \lambda (*)$

Deletion: $a \rightarrow \lambda (r)$



Insertion: $\lambda \rightarrow a (l)$
 $\lambda \rightarrow b (l)$

Insertion: $\lambda \rightarrow a (*)$

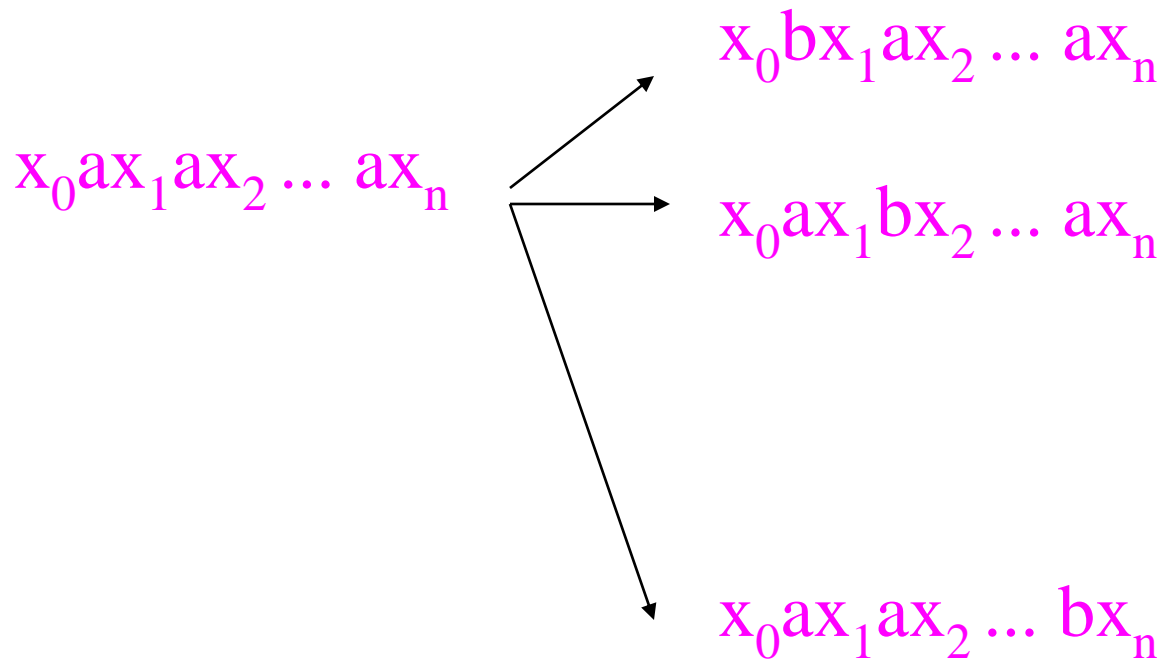


FORMAL DEFINITIONS (1)

EVOLUTIONARY OPERATIONS AND ACTIONS:

Substitution: $a \rightarrow b$

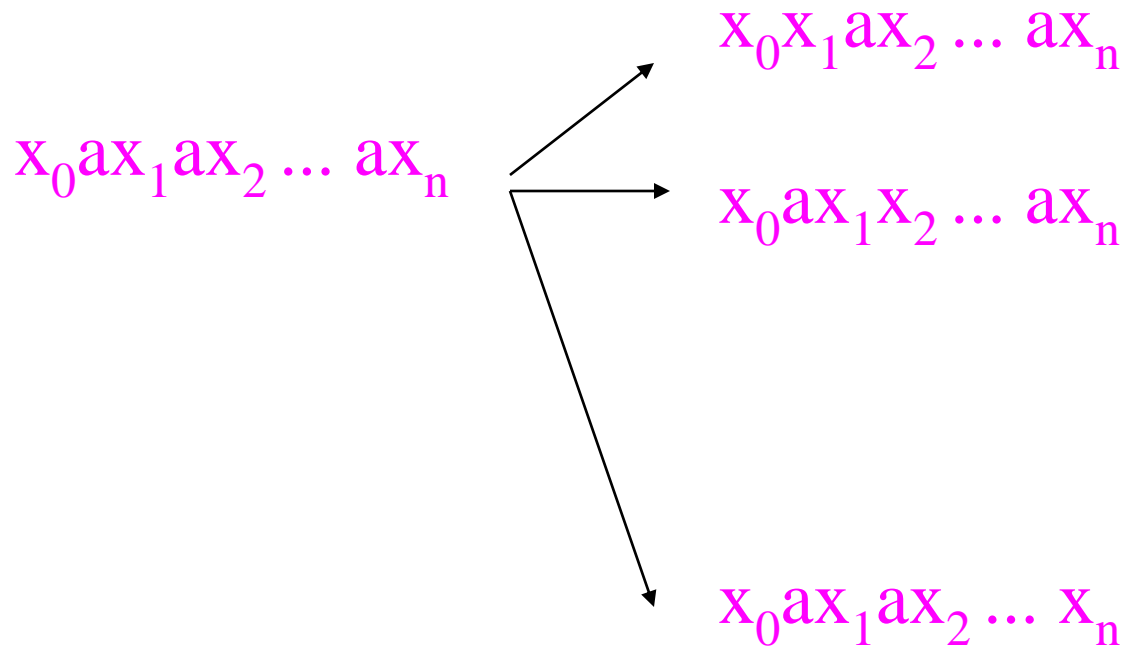
$$\sigma^*(w) = \{ubv : \exists u, v \in V^* (w = uav)\} \text{ or } \sigma^*(w) = \{w\}$$



FORMAL DEFINITIONS (2)

Deletion: $a \rightarrow \lambda$

- $\sigma^*(w) = \{uv : \exists u, v \in V^* (w = uav)\}$ or $\sigma^*(w) = \{w\}$
- $\sigma^r(w) = \{u : w = ua\}$ or $\sigma^r(w) = \{w\}$
- $\sigma^l(w) = \{v : w = av\}$ or $\sigma^l(w) = \{w\}$



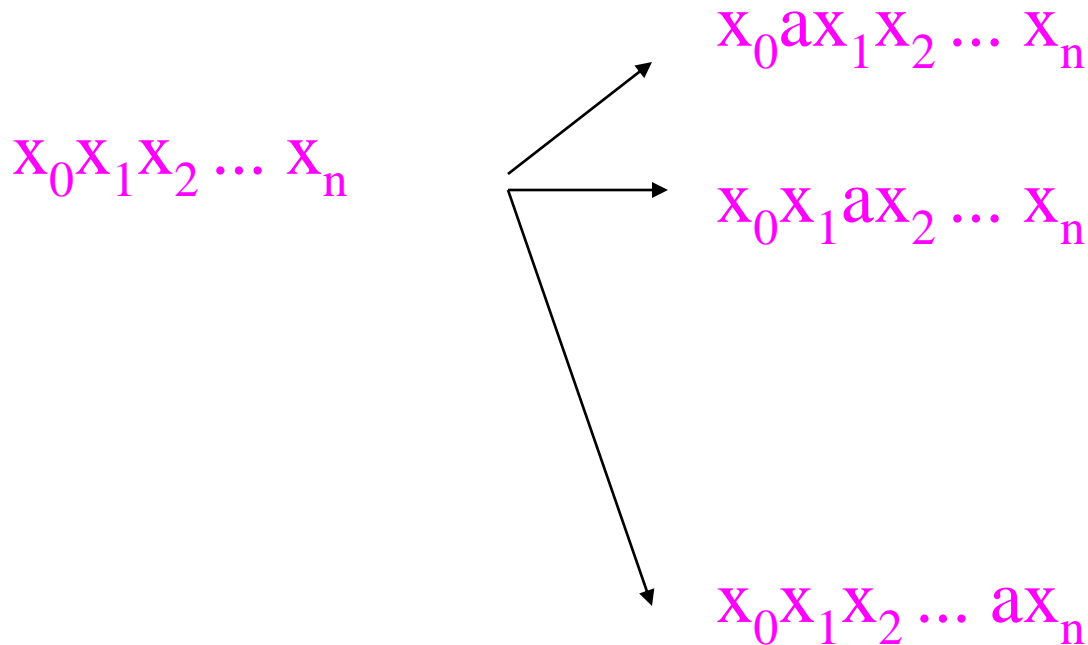
FORMAL DEFINITIONS (3)

Insertion: $\lambda \rightarrow a$

$$\sigma^*(w) = \{uav : \exists u, v \in V^* (w = uv)\}$$

$$\sigma^r(w) = \{wa\},$$

$$\sigma^l(w) = \{aw\}.$$



FORMAL DEFINITIONS (4)

FILTERS:

$$\begin{aligned}\varphi^{(s)}(w;P,F) &\equiv P \subseteq \mathit{alph}(w) \quad \wedge \quad F \cap \mathit{alph}(w) = \emptyset. \\ \varphi^{(w)}(w;P,F) &\equiv \mathit{alph}(w) \cap P \neq \emptyset \quad \wedge \quad F \cap \mathit{alph}(w) = \emptyset.\end{aligned}$$

$$\varphi^{\beta}(L,P,F) = \{w \in L : \varphi^{\beta}(w;P,F)\}.$$

FORMAL DEFINITIONS (5)

EVOLUTIONARY PROCESSOR: (M, PI, FI, PO, FO)

ANEP: $\Gamma = (V, U, G, \mathcal{N}, \alpha, \beta, x_I, x_O)$

$G = (X_G, E_G)$: underlying graph structure

$\mathcal{N} : X_G \rightarrow EP_V$: associated evolutionary processors

$\alpha : X_G \rightarrow \{*, l, r\}$: action mode

$\beta : X_G \rightarrow \{s, w\}$: filter type

$\rho_x(\cdot) = \varphi^{\beta(x)}(\cdot; PI_x, FI_x)$: input filter

$\tau_x(\cdot) = \varphi^{\beta(x)}(\cdot; PO_x, FO_x)$: output filter

FORMAL DEFINITIONS (6)

WORKING MODE

Evolutionary step:

$$C \Rightarrow C', \text{ iff } C'(x) = M_x^\alpha(x)(C(x))$$

Communication step:

$$C \triangleright\triangleright C' \text{ iff}$$

$$C'(x) = (C(x) - \tau_x(C(x))) \cup \bigcup_{\{x,y\} \in EG} (\tau_y(C(y)) \cap \rho_x(C(y)))$$

ACCEPTED/DECIDED LANGUAGE

1. **There exists a configuration in which the set of words existing in the output node x_o is non-empty.**
2. **There exist two consecutive identical configurations.**
3. **It works forever.**

$L_{acc}(\Gamma) = \{w \in V^* : \text{the computation of } \Gamma \text{ on } w \text{ is an accepting one}\}.$

L is decided by Γ iff $L_{acc}(\Gamma) = L$ and Γ halts on every input

COMPUTABILITY COMPLETENESS

Th1. There class of languages accepted by ANEPs is exactly the class of recursively enumerable languages.

Th2. The class of recursive languages is included in the class of languages decided by ANEPs.

Constant size: 31

UNIVERSALITY

There exists Γ_U with the input alphabet A such that on any input $\langle \Gamma \rangle \langle w \rangle$:

- Γ_U halts on $\langle \Gamma \rangle \langle w \rangle$ if and only if Γ halts on w .**
- $\langle \Gamma \rangle \langle w \rangle$ is accepted by Γ_U if and only if w is accepted by Γ .**

Moreover, $\text{size}(\Gamma_U) = 5|A| + 8$.

TIME COMPLEXITY

The *time complexity* of the halting computation $C_0(x), C_1(x), C_2(x), \dots, C_m(x)$ of Γ on $x \in L$ is denoted by $Time_\Gamma(x)$ and equals m .

$$Time_\Gamma(n) = \max\{Time_\Gamma(x) : x \in L_{acc}(\Gamma), |x|=n\}.$$

$Time_{ANEP}(f(n)) = \{L : L = L_{dec}(\Gamma) \text{ for an ANEP } \Gamma \text{ with } Time_\Gamma(n) \leq f(n) \text{ for some } n \geq n_0\}.$

$$PTime_{ANEP} = \bigcup_{k \geq 0} Time_{ANEP}(n^k).$$

SPACE COMPLEXITY

The *space complexity* of the halting computation $C_0^{(x)}, C_1^{(x)}, C_2^{(x)}, \dots, C_m^{(x)}$ of Γ on $x \in L$ is denoted by $Space_{\Gamma}(x)$.

$Space_{\Gamma}(n) = \max\{\max\{\max\{|C_i^{(x)}(z)| : z \in X_G\} : i=1..n\} : |x|=n\}$.

$Space_{ANEP}(f(n)) = \{L : L = L_{dec}(\Gamma) \text{ for an ANEP } \Gamma \text{ with } Space_{\Gamma}(n) \leq f(n) \text{ for some } n \geq n_0\}$.

$PSPACE_{ANEP} = \bigcup_{k \geq 0} Space_{ANEP}(n^k)$.

RELATIONSHIPS (1)

Proposition

For any Turing machine M deciding a language L there exists a complete/star ANEP Γ deciding the same language L . Furthermore, $\text{Time}_{\Gamma}(n) \leq 22T_M(n)$.

Remarks:

- $\text{Size}(\Gamma) = 18 + 7(\text{card}(U) - 1) + \text{card}(Q) + (\text{card}(U) - 1)^2 + 2\text{card}(Q)(\text{card}(U) - 1)$.**
- $\text{Symb}(\Gamma) = 4\text{card}(Q)(\text{card}(U) - 1)^2 + 2(\text{card}(U) - 1)^2 + 2\text{card}(Q)(\text{card}(U) - 1) + 10(\text{card}(U) - 1) + \text{card}(Q)$.**

RELATIONSHIPS (2)

Theorem.

$$NP = PTime_{ANEP}$$

M chooses **nondeterministically** a copy of the input word from those existing in the initial node of Γ and follows its itinerary through the underlying network of Γ .

It takes the quadratic time of Γ

Theorem.

$$NP = PTime_{ANEP(24)}$$

RELATIONSHIPS (3)

Theorem. A language L is in P iff there exists an ANEP Γ and two polynomials $p(n)$ and $q(n)$ such that

1. $L = L_{dec}(\Gamma)$.
2. $Space_{\Gamma}(n) \leq q(n)$ and $Time_{\Gamma}(n) \leq p(n)$.

$$PTime_{ANEP} \cap PSpace_{ANEP} \subseteq P$$

FALS

SOLVING PROBLEMS WITH ANEPs(1)

A decision problem P is solved in time $O(f(n))$ by ANEPs if there exists a family \mathcal{A} of ANEPs such that:

1. The encoding function of any instance p of P having size n can be computed by a deterministic Turing machine in time $O(f(n))$.
2. For each instance p of size n of the problem one can effectively construct, in time $O(f(n))$, an ANEP $\Gamma(p) \in \mathcal{A}$ which decides, again in time $O(f(n))$, the word encoding the given instance.

If an ANEP $\Gamma \in \mathcal{A}$, constructed as above, decides the language of words encoding all instances of the same size n , then the construction of \mathcal{A} is called a **uniform solution**.

SOLVING PROBLEMS WITH ANEPs(2)

NP-complete problems solved by complete/star ANEPs

Problem	Time	Size	Other
3-COL	$O(m+n)$	$O(m+n)$	$O(n+n)$
BPCP	$O(nm)$	$O(nm)$	$O(nm)$
CAP	$O(m+n)$	$O(m+n)$	$O(m+n)$
3CNF-SAT	$O(m+n)$	$O(m+n)$	$O(m+n)$
HPP	$O(n)$	$O(n)$	$O(n)$

COMMON ALGORITHMIC PROBLEM

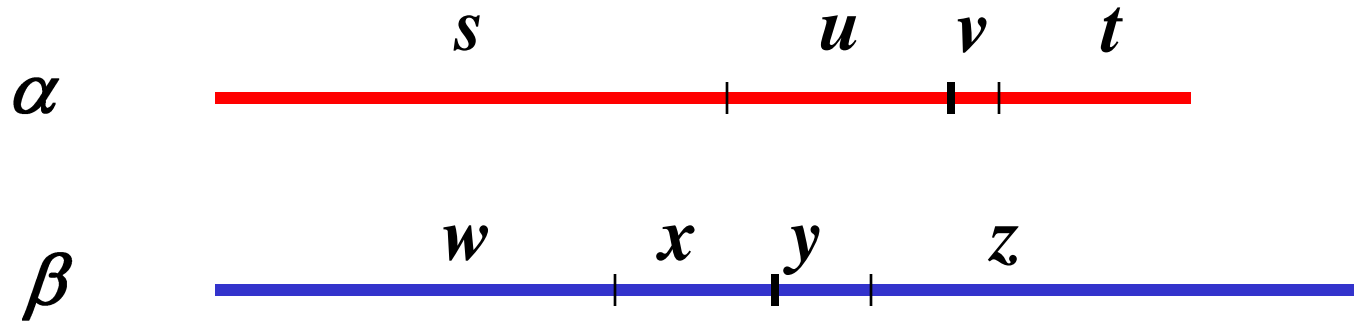
Let S be a finite set and F be a non-empty family of subsets of S . Find the cardinality of a maximal subset of S which does not include any set belonging to F .

The maximal independent set problem

The vertex cover problem

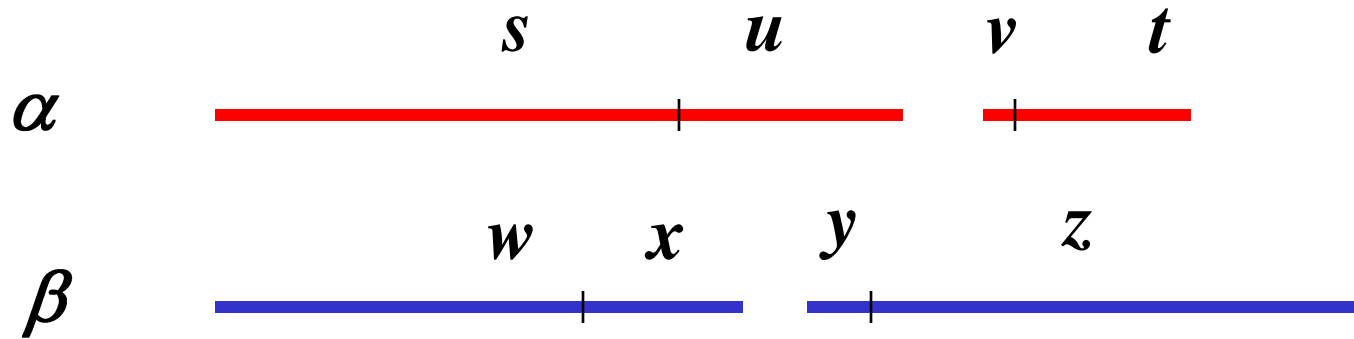
The satisfiability problem

SPLICING(1)



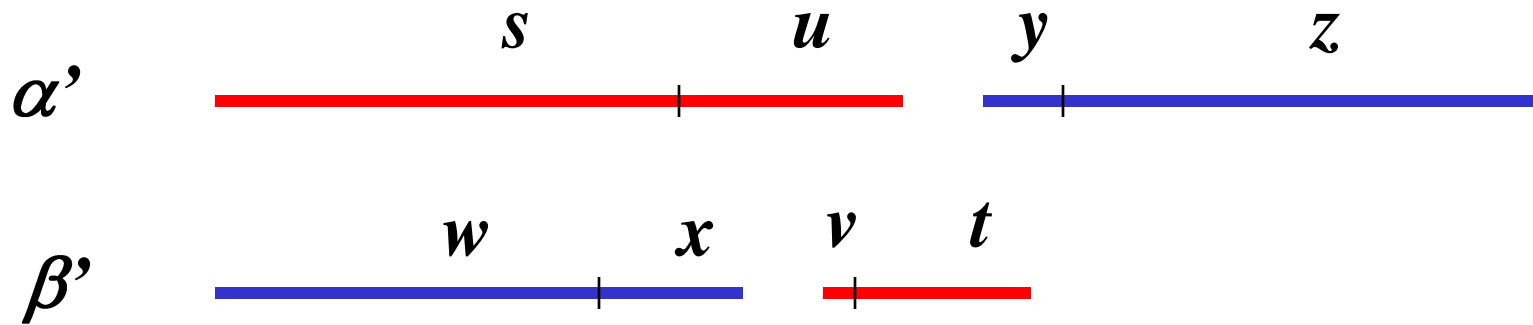
$$\sigma = [(u, v); (x, y)]$$

SPLICING(2)



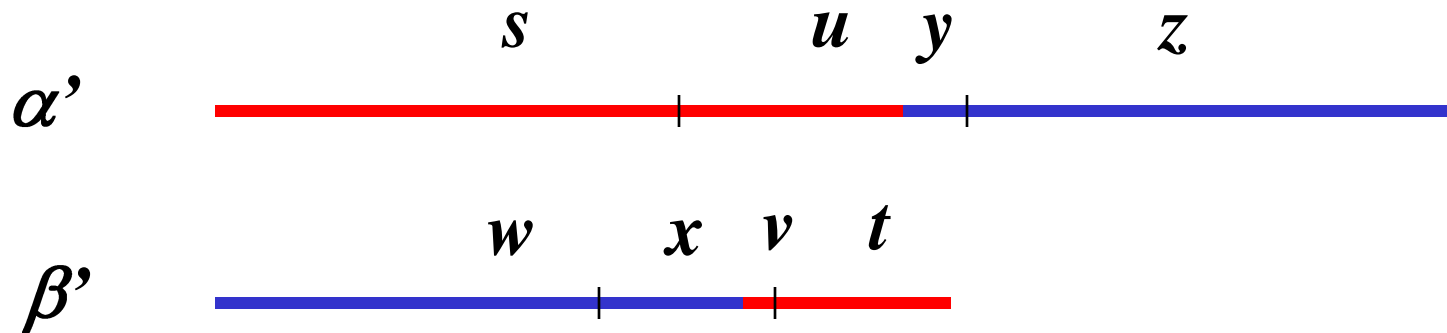
$$\sigma = [(u, v); (x, y)]$$

SPLICING(3)



$$\sigma = [(u, v); (x, y)]$$

SPLICING(4)



$$\sigma(\alpha, \beta) = \{\alpha'\} \cup \{\beta'\}$$

$$\sigma(L) = \bigcup_{x, y \in L} \sigma(x, y)$$

$$M(L) = \bigcup_{\sigma \in M} \sigma(L)$$

ANSP

SPLICING PROCESSOR: (S, A, PI, FI, PO, FO)

ANSP: $\Gamma = (V, U, <, >, G, \mathcal{N}, \alpha, x_I, x_O)$

COMPUTING WITH ANSPs

Theorem. ANSPs with **three** nodes can simulate any NTM.

Theorem : $NP = PTime_{ANSP(3)}$

Theorem: $P \subseteq PTime_{ANSP(2)}$

2=3 \equiv P=NP ?

A FUNNY (IM)POSSIBLE IMPLEMENTATION

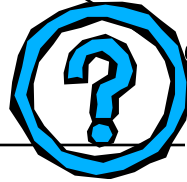
Output node



Substitution: $a \rightarrow b$
 $a \rightarrow c$

Deletion: $a \rightarrow \lambda (*)$
 $b \rightarrow \lambda (*)$

Deletion: $a \rightarrow \lambda (r)$



Insertion: $\lambda \rightarrow a (l)$
 $\lambda \rightarrow b (l)$



Insertion: $\lambda \rightarrow a (*)$



Thank You