Coq 証明支援系による DNA 計算の形式化

TPP2024　　11/26

九州大学　数理学府

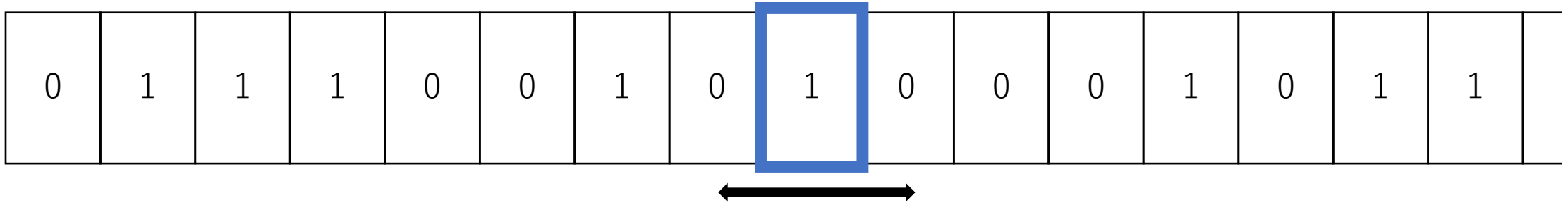
早川　銀河

目次

- DNA計算とは
- DNAドミノ
- スティッカーシステム
- オートマトンとスティッカーシステムの関係
- SSReflect/finType
- Coqでの実装
- まとめと今後の課題

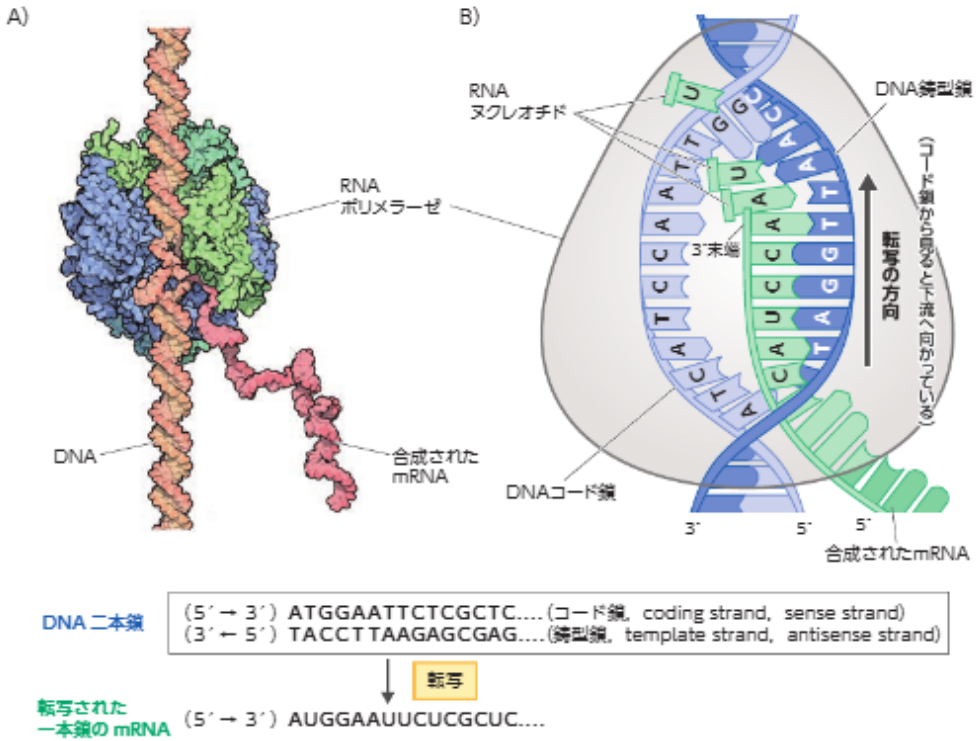
DNA計算とは　(1)チューリングマシン

チューリングマシンは無限長のテープとテープ上を移動して読み書きを行うヘッドからなる、概念上の計算機である。

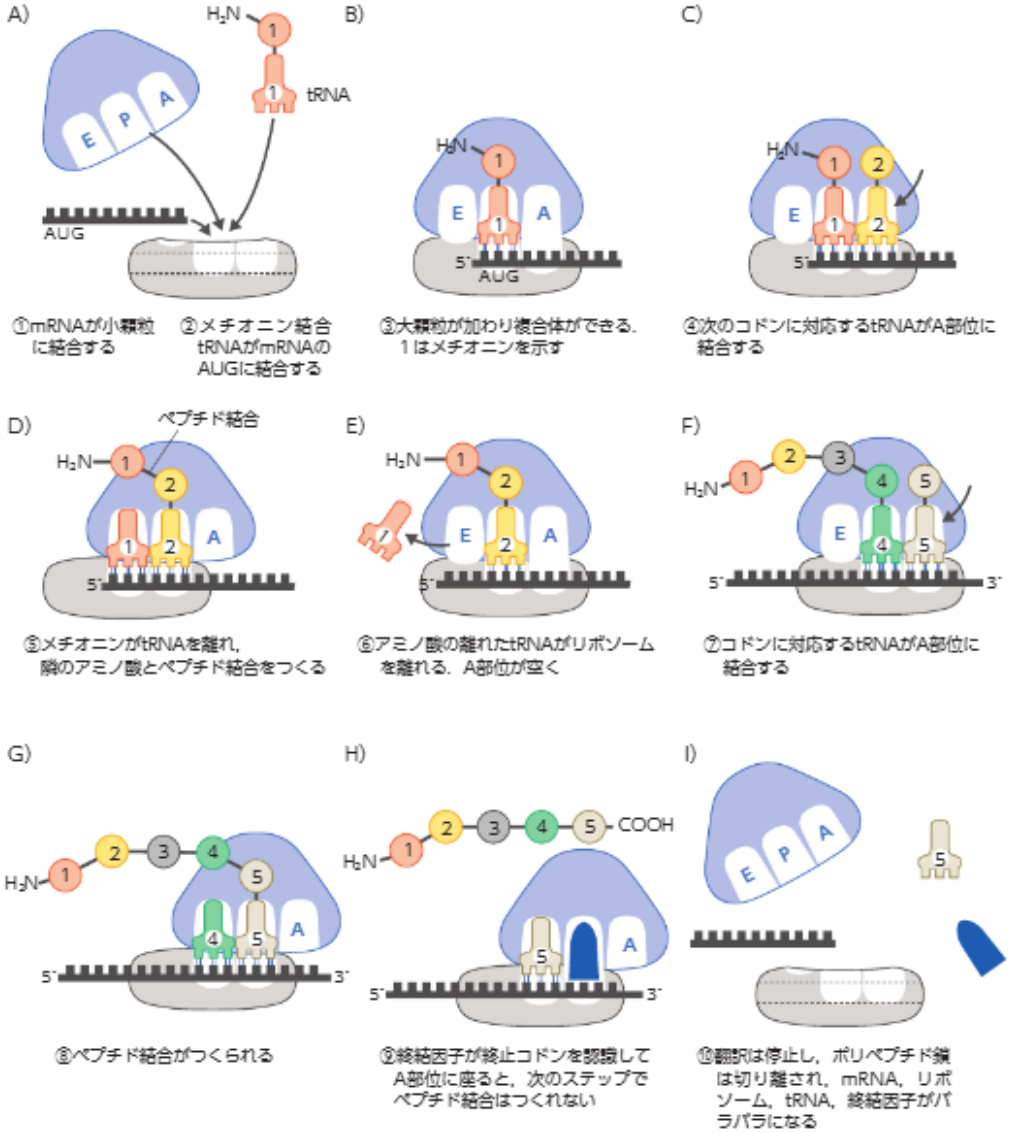


チューリングが1936年に提唱し、その型を参考にしてノイマン型コンピュータが開発された。

DNA計算とは (2)
人体が持つ“計算機”



●図3-12　RNAポリメラーゼによるmRNAの転写
A) はApril 2003 Molecule of the Month by David Goodsell より，B) は参考図書1より作成



●図3-20　翻訳の過程
A〜E) 開始，F〜G) 伸長，H〜I) 終了．開始コドンの上流には本来は5′非翻訳領域があるが省略している．参考図書3をもとに作成

基礎から学ぶ生物学・細胞生物学 第4版 P93

DNADomino

- DNA の二本鎖構造を参考にして設計された分子装置
- 中央部分で相補的に結合する二本鎖と、両端に露出した粘着末端からなる

$$\left(\binom{V^*}{\lambda} \cup \binom{\lambda}{V^*}\right) \left[\begin{matrix} V^+ \\ V^+ \end{matrix}\right]_\rho \left(\binom{V^*}{\lambda} \cup \binom{\lambda}{V^*}\right)$$

V：アルファベット

V^*：クリーネ閉包(+は非空)

λ：空文字

$\rho \subseteq V \times V$：相補的対応関係

StickerSystem

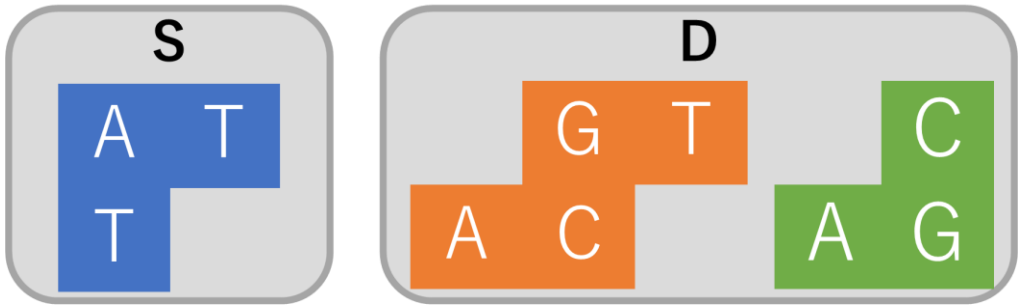
スティッカーシステム：$\gamma = (V, \rho, S, D)$
V：アルファベット（記号の有限集合）　　$\rho \subseteq V \times V$
S：二本鎖部を持つドミノの有限集合　　D：ドミノ対の有限集合

SのドミノにDのドミノ対を繰り返し結合させていき、生成されるドミノを文字列に変換することでスティッカーの生成する言語とする。

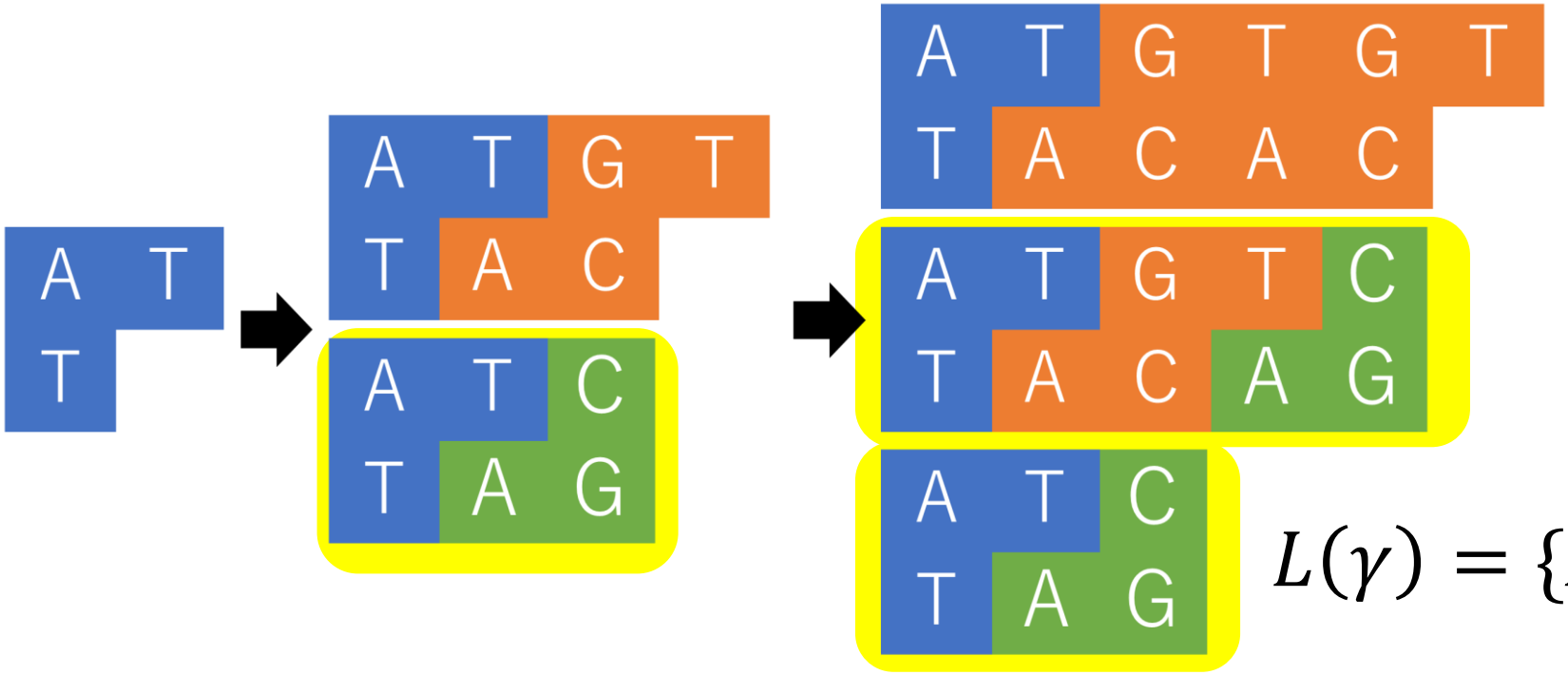
スティッカーシステムの生成言語

$$L(\gamma) = \left\{ w \in V^* \middle| \begin{bmatrix} w \\ w' \end{bmatrix}_\rho \in (\gamma\text{が生成するドミノの集合}) \right\}$$

StickerSystem



$$V = \{A, T, G, C\}$$
$$\rho = \{(A,T), (T,A), (G,C), (C,G)\}$$
$$S = \left\{\begin{bmatrix} A \\ T \end{bmatrix}\begin{pmatrix} T \\ \lambda \end{pmatrix}\right\}$$
$$D = \left\{\left(\begin{pmatrix} \lambda \\ \lambda \end{pmatrix}, \begin{pmatrix} \lambda \\ A \end{pmatrix}\begin{bmatrix} G \\ C \end{bmatrix}\begin{pmatrix} T \\ \lambda \end{pmatrix}\right), \left(\begin{pmatrix} \lambda \\ \lambda \end{pmatrix}, \begin{pmatrix} \lambda \\ A \end{pmatrix}\begin{bmatrix} C \\ G \end{bmatrix}\right)\right\}$$
$$\gamma = (V, \rho, S, D)$$



$$L(\gamma) = \{AT(GT)^n C | n \in \mathbb{N}\}$$

Automaton vs StickerSystem

(右側)スティッカーシステムは有界遅延の原始的な計算で正則文法と同じ生成能力を持つことが知られている。

$REG = RSL$　(REG: 正則言語, RSL: 右側スティッカーの生成言語)

今回、オートマトンを模倣するスティッカーシステムを構成することで

$REG \subseteq RSL$を形式的に示した。

```
Theorem REG_RSL{state symbol:finType}(M:@automaton state symbol)(s:seq symbol)
(m:nat):s <> nil -> exists n:nat , n <= m ->
accept M s = (s\in(ss_language_prime m (Aut_to_Stk M))).
```

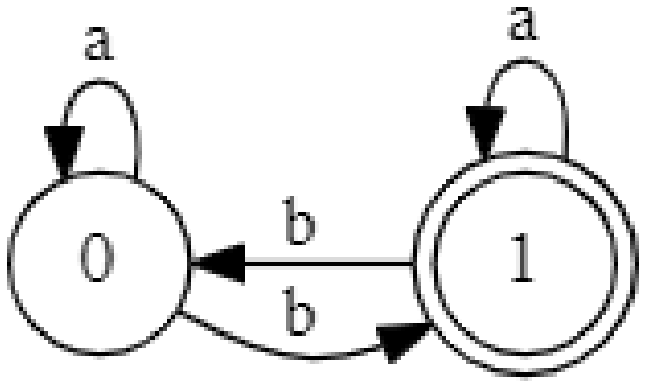
Automaton

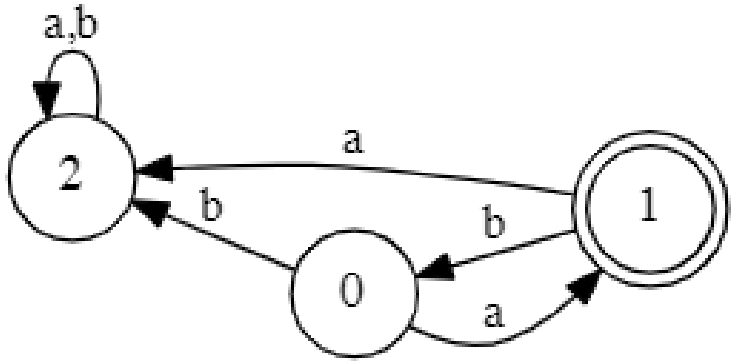
有限オートマトン $\mathrm{M} = (K, V, s_0, F, \delta)$

K:状態の有限集合　　V: アルファベット

$\delta: K \times V \to K$: 遷移関数　　$s_0 \in K$:初期状態　$F \subseteq K$:受理状態

オートマトンの受理言語 $L(M) = \{V^* | \delta^*(s_0, V^*) \in F\}$





Coq/SSReflect/finType

有限の要素のみからなる型

eqType,countType等の特徴を引き継いでいる

enumにより要素の全列挙が可能となる

(例)enum bool_finType = [::true;false]

オートマトンやスティッカーシステムで扱う文字、文字列をascii,stringの代わりにfinTypeとそのリストとすることで、より厳密な定義が可能となる.

オートマトンの実装

有限オートマトン $\mathrm{M} = (K, V, s0, F, \delta)$
K:状態の有限集合, V: アルファベット(文字の有限集合)
$\delta: K \times V \rightarrow K$: 遷移関数 $s_0 \in K$初期状態, $F \subseteq K$受理状態

```
Structure automaton{state symbol:finType}:= Automaton {
  init : state;
  final : {set state};
  delta : state -> symbol -> state
}.
```

オートマトンの実装

$\delta: (K, V\ ) \rightarrow K$: 遷移関数
$\delta^*: K \times V^* \rightarrow K$:遷移関数の推移的反射閉包
オートマトンの受理判定:$\delta^*(s_0, w)\ \in F\ or \notin F \qquad (w \in V^*)$

```
Fixpoint dstar{state symbol:finType}(delta:state->symbol->state)
  (q:state)(str:seq symbol):state :=
match str with
|nil => q
|h::str' => dstar delta (delta q h) str'
end.
```

```
Definition accept{state symbol:finType}(M:@automaton state symbol)
  (str:seq symbol):bool := dstar (delta M) (init M) str\in final M.
```

DNAドミノの実装

中央の二本鎖部と末端の一本鎖部をそれぞれ定義し，その組み合わせで様々なドミノの形状を表現する

```
Inductive domino{symbol:finType}{rho:Rho symbol}:=
|null : domino
|Simplex : @stickyend symbol -> domino
|WK : @wk symbol rho -> domino
|L : @stickyend symbol -> @wk symbol rho -> domino
|R : @wk symbol rho -> @stickyend symbol -> domino
|LR : @stickyend symbol -> @wk symbol rho -> @stickyend symbol -> domino.
```

スティッカーシステムの実装

スティッカーシステム：$\gamma = (V, \rho, S, D)$

V：アルファベット（記号の有限集合）　$\rho \subseteq V \times V$

S：二本鎖部を持つドミノの有限集合　　D：ドミノ対の有限集合

```
Structure sticker{symbol : finType}{rho : seq(symbol*symbol)} :={
  start : seq (@domino symbol rho);
  extend : seq ((@domino symbol rho)*(@domino symbol rho))
}.
```

Automaton vs StickerSystem

```
Theorem REG_RSL{state symbol:finType}(M:@automaton state symbol)(s:seq symbol)
(m:nat):s <> nil -> exists n:nat , n <= m ->
accept M s = (s\in(ss_language_prime m (Aut_to_Stk M))).
```

まとめと今後の課題

今回、(片側)スティッカーシステムが正則文法以上の生成能力を持つことをCoqで形式的に検証した。

DNA計算モデルは他にも多数存在し、中には帰納的加算言語を生成可能なものも存在する。

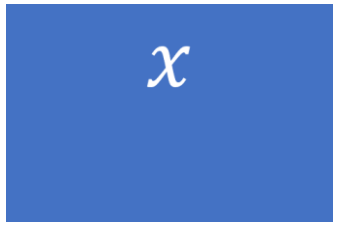
- (両側)スティッカーシステム
- ワトソンクリックオートマトン
- 挿入・削除システム
- スプライシングシステム
- 環状スプライシングシステム

今後、これらのモデルに関する証明も形式化していきたい。

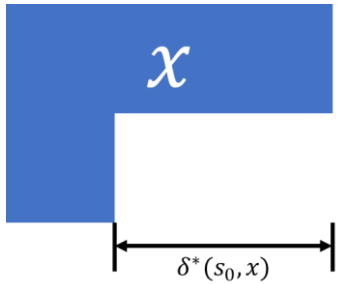
ご清聴ありがとうございました。

Automaton vs StickerSystem

S
(開始ドミノ)

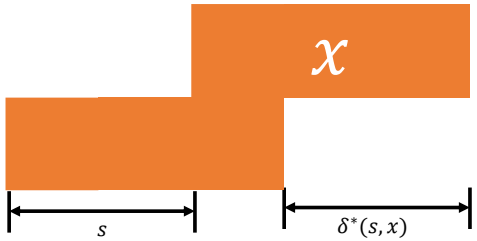


オートマトンがxを受理する
$\delta^*(s_0, x) \in F$

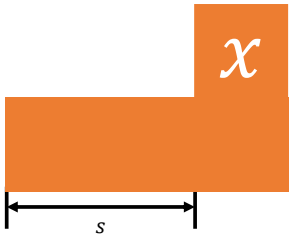


オートマトンにxを作用させた状態
$\delta^*(s_0, x)$を粘着末端で表現

D(結合させる
ドミノ)

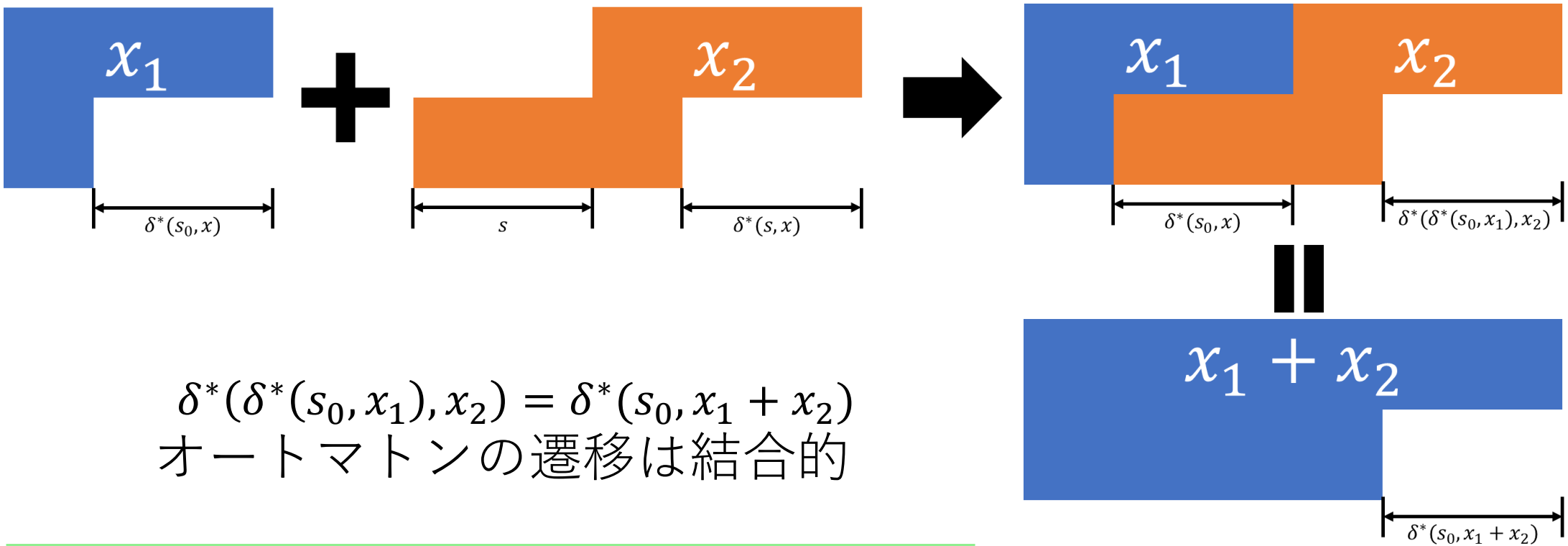


オートマトンがある状態sからxにより
遷移した状態$\delta^*(s, x)$を粘着末端で表現



オートマトンがある状態sから
xを受理する　$\delta^*(s, x) \in F$

Automaton vs StickerSystem

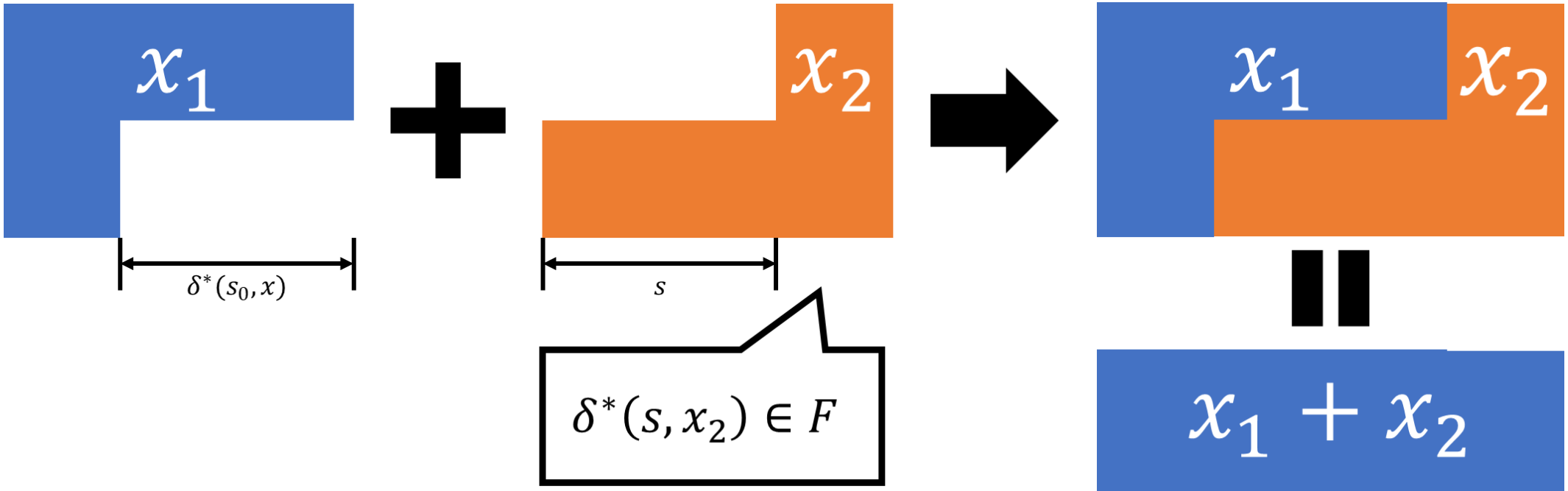


$$\delta^*(\delta^*(s_0, x_1), x_2) = \delta^*(s_0, x_1 + x_2)$$

オートマトンの遷移は結合的

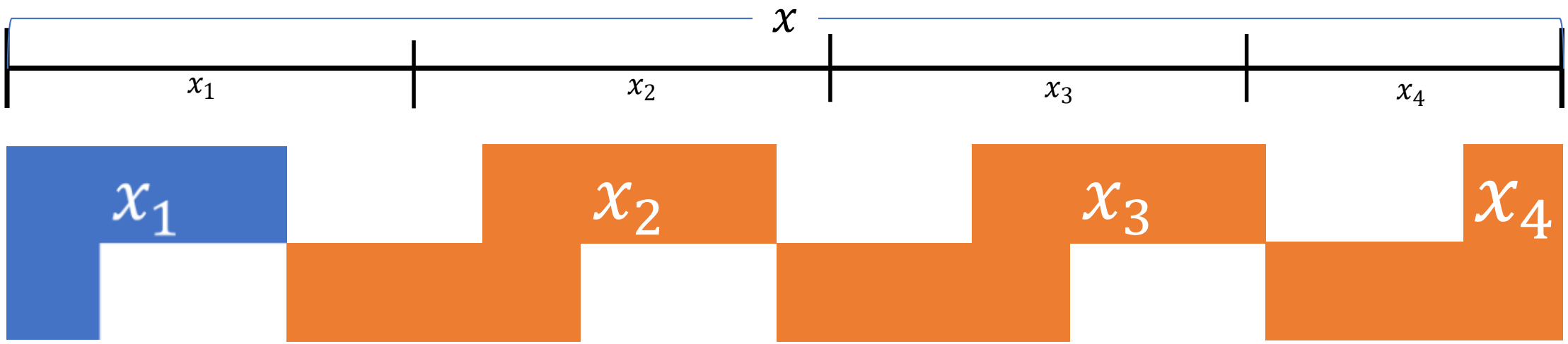
```
Lemma mu'lemma{state symbol:finType}(M:@automaton state symbol)
(s t u:seq symbol):#|state|.+1<=size s -> size t = #|state|.+1 -> u<>nil->
mu' (startDomino M s)(extentionDomino M t u)=
if drop(size s-((index(dstar(delta M)(init M)s)(enum state)).+1))s == u then
  Some (startDomino M (s++t))
else
  None.
```

Automaton vs StickerSystem




```
Lemma mu'lemma2{state symbol:finType}(M:@automaton state symbol)
(s t u:seq symbol)(d:domino*domino): #|state|<size s ->
Some d = (stopDomino M u t)->
mu'(startDomino M s)d =
if (drop (size s-(index(dstar(delta M)(init M)s)(enum state)).+1)s)
  ==u then
  mkWK (s++t)
else
  None.
```

Automaton vs StickerSystem



$$\delta^*(\delta^*(\delta^*(\delta^*(s_0, x_1), x_2), x_3), x_4) \in F \quad \Leftrightarrow \delta^*(s_0, x_1 + x_2 + x_3 + x_4)$$
$$\Leftrightarrow \delta^*(s_0, x) \in F \Leftrightarrow x\text{はオートマトンにより受理される}$$

任意の文字列は分割した有限長の文字列に対応するドミノの
組み合わせで受理判定が可能
⇒オートマトンの受理判定をスティッカーで模倣できる

付録　チョムスキー階層

ノーム・チョムスキーは言語を生成文法と対応付け、その包含関係で階層分けした。(1956)

各階層はオートマトンなどの受理機械によっても特徴づけられることが知られている。

チョムスキー階層	言語	受理機械
Type-0	帰納的加算	チューリングマシン
Type-1	文脈依存	線型拘束オートマトン
Type-2	文脈自由	プッシュダウンオートマトン
Type-3	正規　正則	（有限）オートマトン