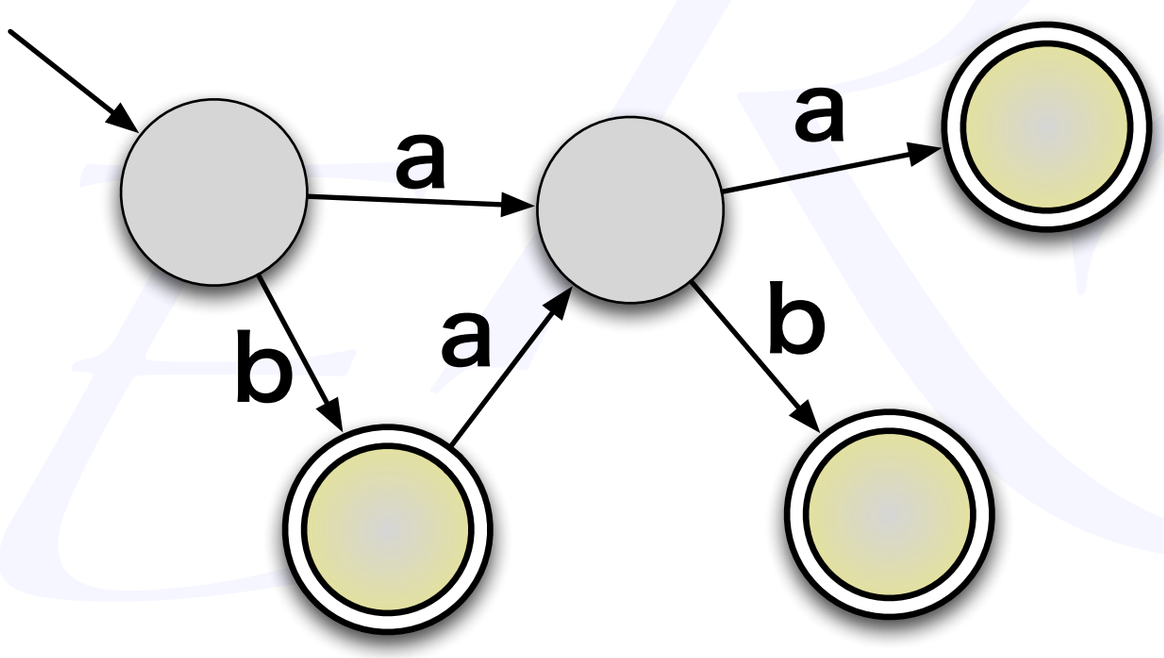
SequenceBDD と文字列集合データ構造

吉仲　亮
科学技術振興機構
ERATO 湊離散構造プロジェクト

決定性有限状態機械と sequenceBDD

決定性有限状態機械

- 文字列の(有限)集合のグラフ表現：
  (無循環)決定性有限状態機械
  (Acyclic Deterministic Finite Automata)



⇒ {b, aa, ab, baa, bab}

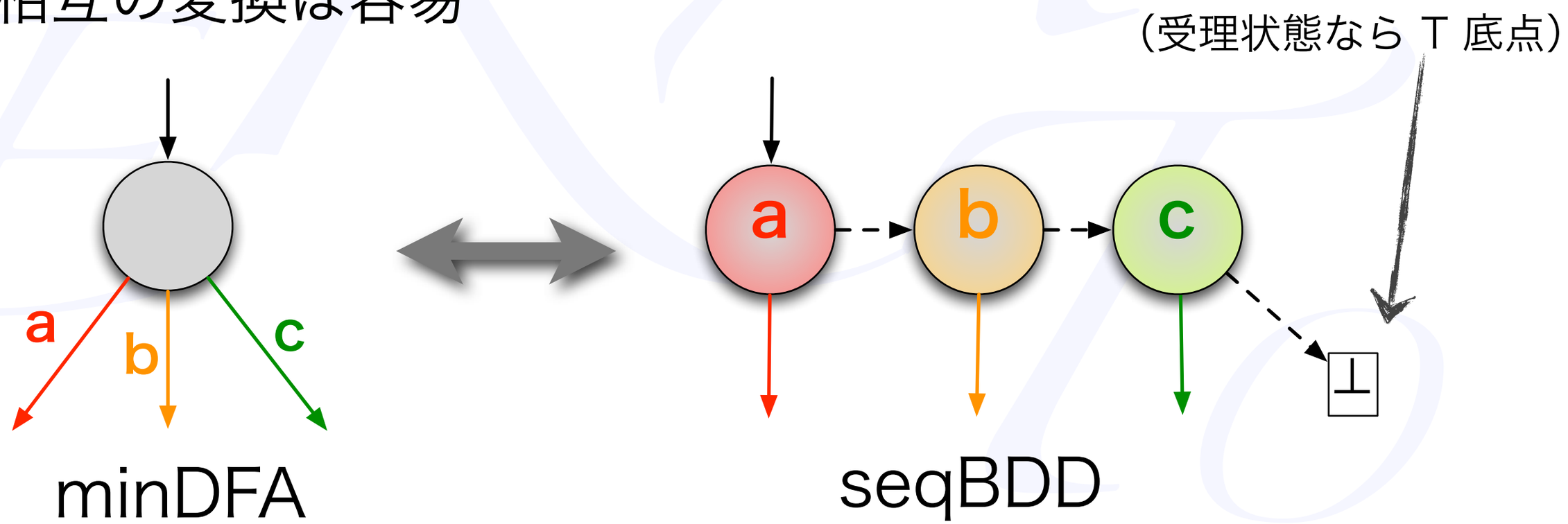
- 任意の正規言語に対して最小のDFA が一意に定まる

Seq-BDD と minDFA

- seq-BDD と minDFA は “ほぼ” 同じ静的構造
- ただし演算処理では seq-BDD は DFA にはない特徴がある
- seq-BDD
  - 各節点の表現する言語は不変
  - １つの言語に対して節点は１つ
    - ２つの言語が等しい＝ポインタが一致　⇒　定数時間判定
- DFA

Minimal DFA vs SequenceBDD

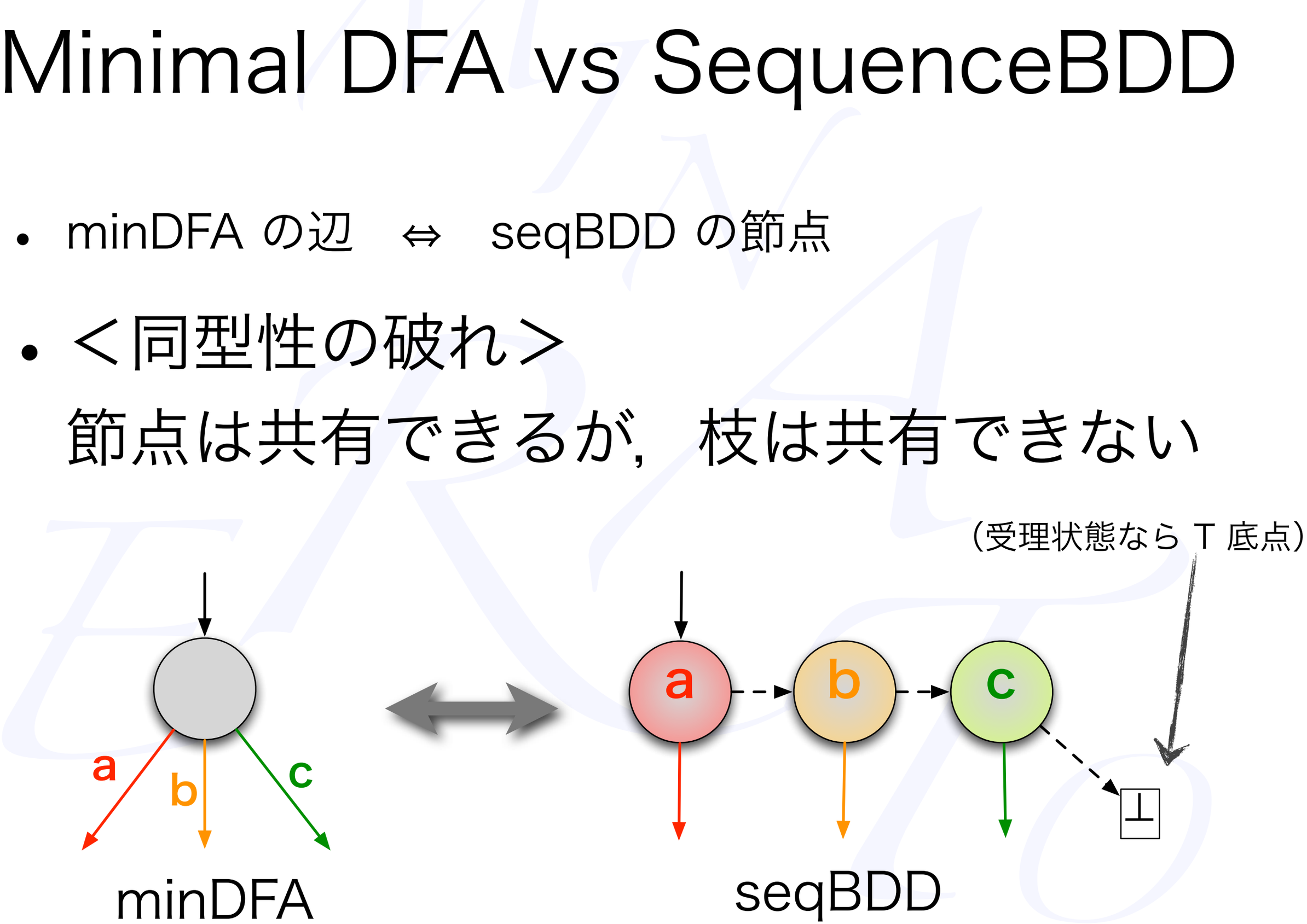
- 有限言語に対して一意な圧縮表現を与える
  （seqBDD では文字順序固定のとき）
- minDFA の辺　⇔　seqBDD の節点
- 相互の変換は容易



Minimal DFA vs SequenceBDD

- minDFA の辺　⇔　seqBDD の節点
- ＜同型性の破れ＞

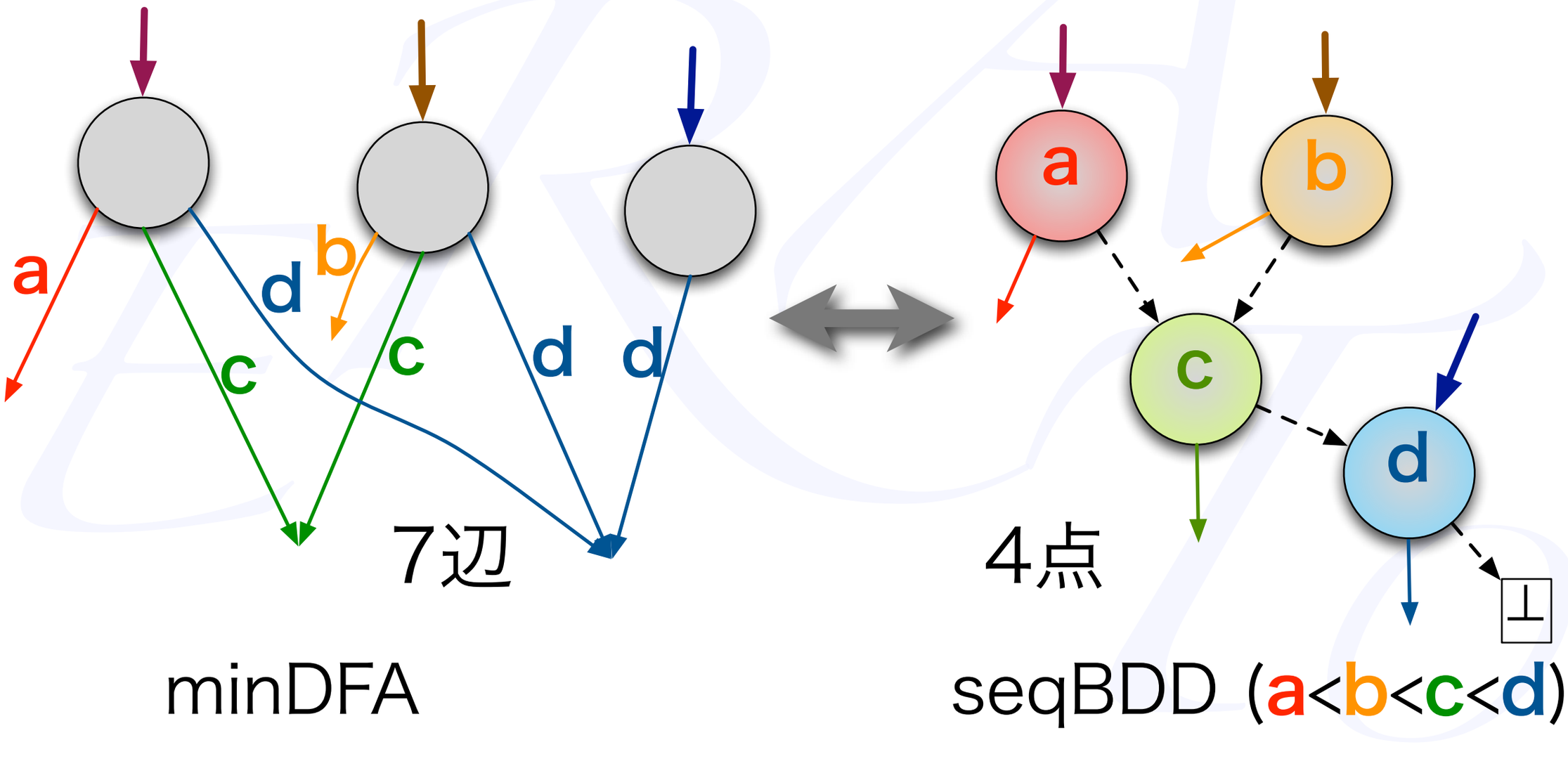
節点は共有できるが，枝は共有できない



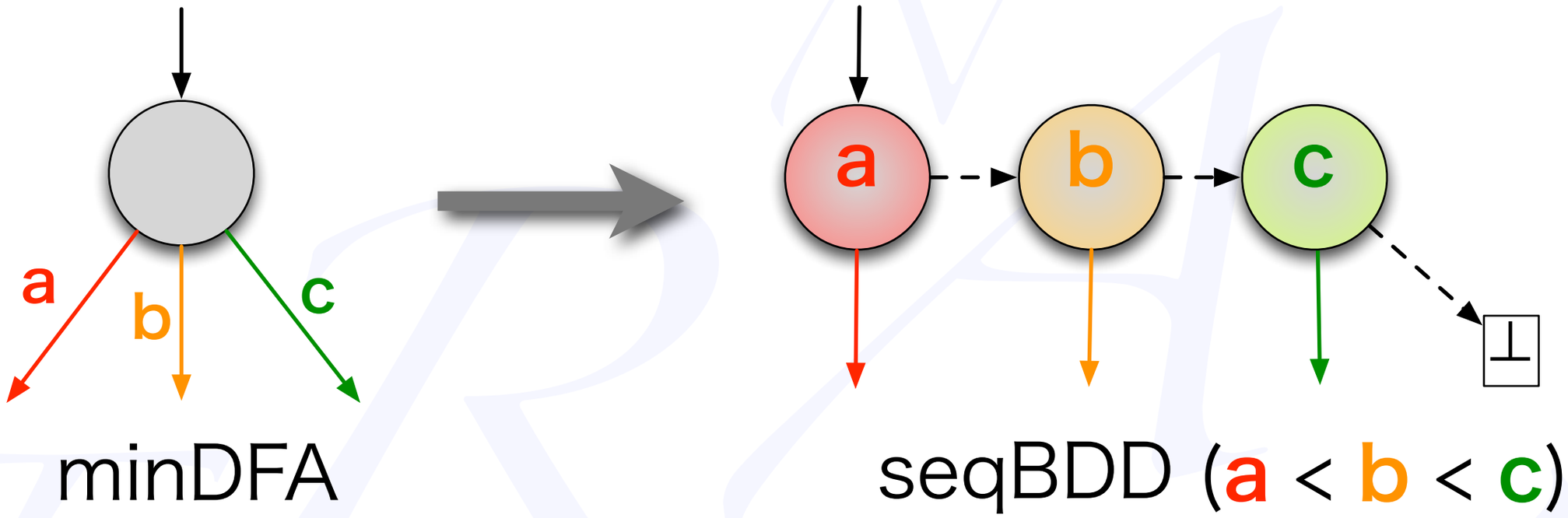
Minimal DFA vs SequenceBDD

- ＜同型性の破れ＞

  節点は共有できるが，辺は共有できない



minDFA to seq-BDD

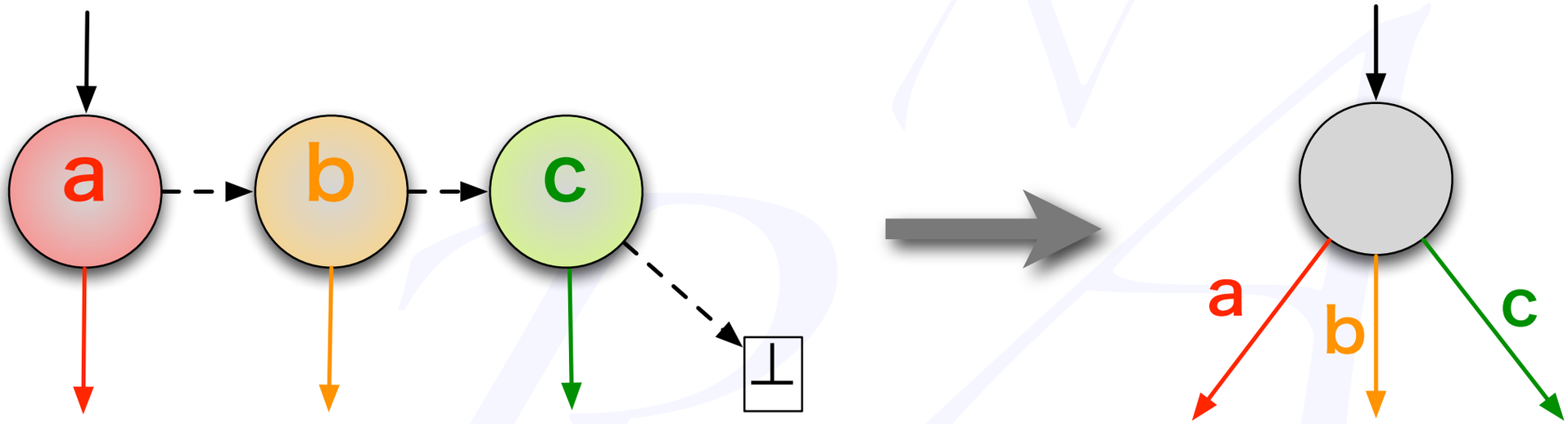


minDFA　　seqBDD (**a** < **b** < **c**)

- [seqDD 節点数] ≦ [minDFA 辺数]
  （ [seqDD 節点数] は底節点数を数えてない ）

- 非既約なseqDDへの変換（線形時間簡約化アルゴリズム）

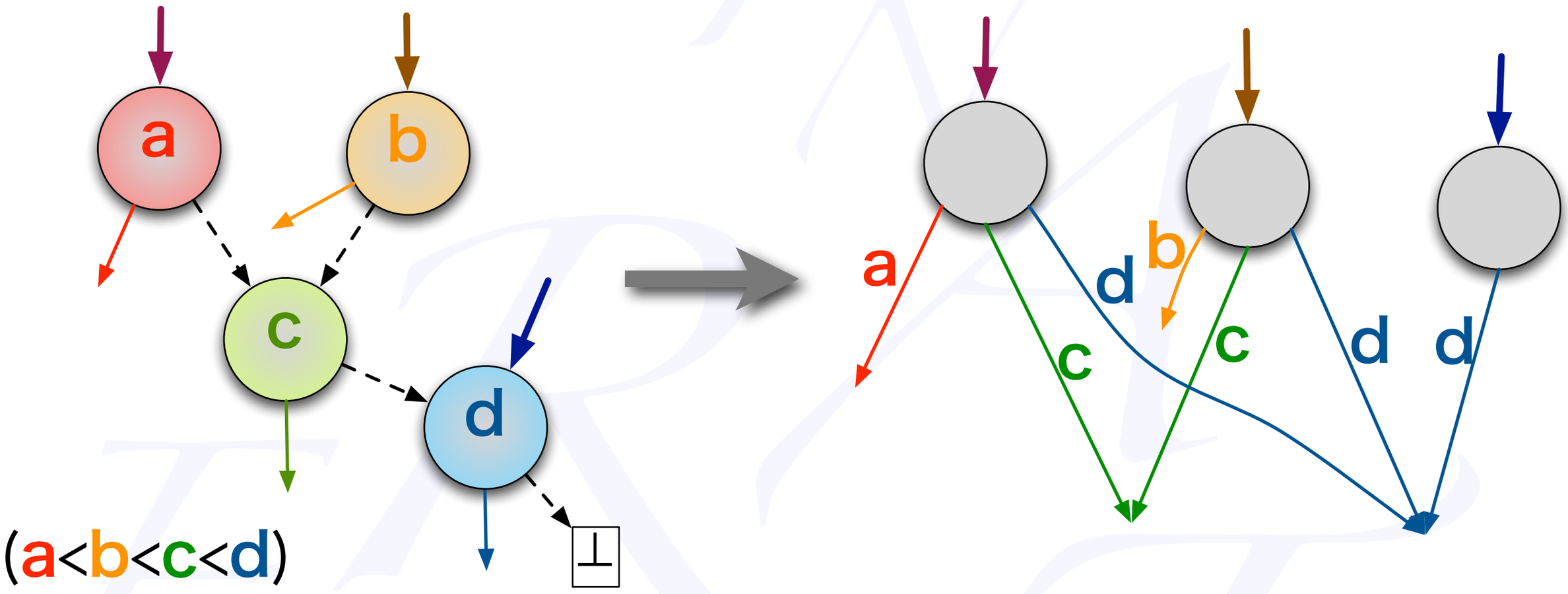
seq-BDD to minDFA



seqBDD (**a**<**b**<**c**)

minDFA

seq-BDD to minDFA

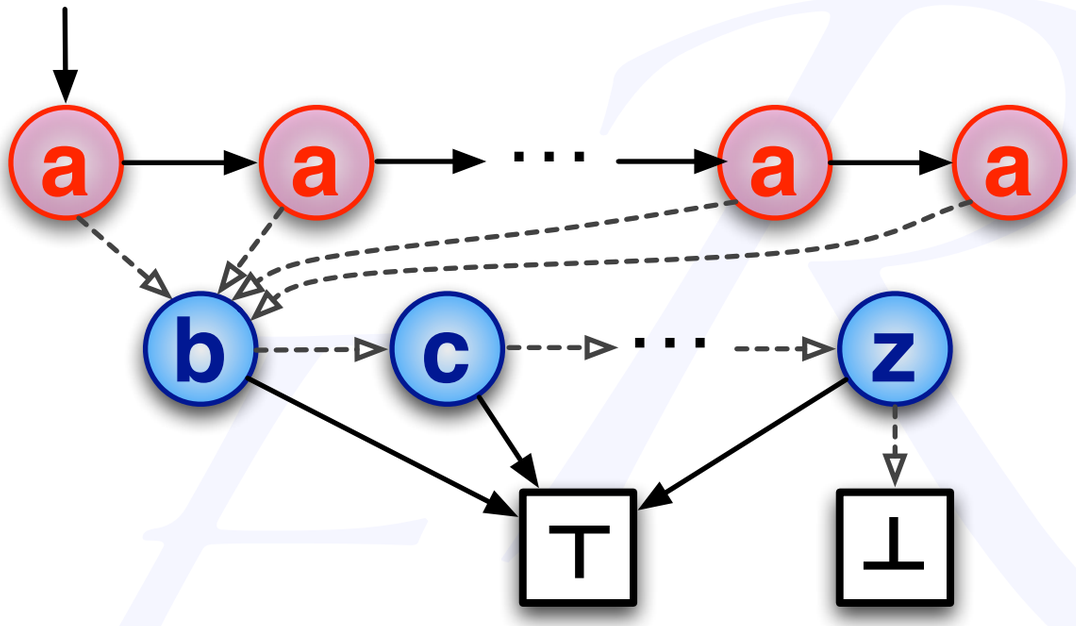


- [minDFA 節点数] ≦ [seqBDD 節点]
  [minDFA 辺数] ≦ [minDFA 節点数]×|Σ|
- [minDFA 辺数] ≦ [seqBDD 節点数]×|Σ|

SequenceBDD vs Minimal DFA

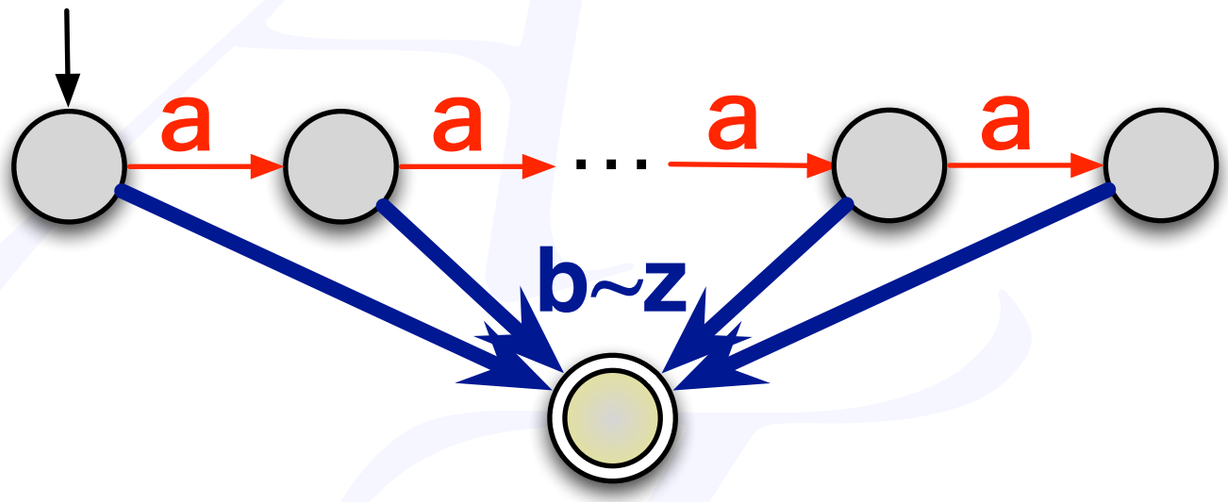
$$L_n = \{\ \mathbf{a}^k\mathbf{x} \mid k \leqq n,\ \mathbf{x} \in \Sigma\backslash\{\mathbf{a}\}\ \}$$

seqBDD (**a**<**b**<**...**<**z**)



節点数：$n+|\Sigma|-1$

minDFA



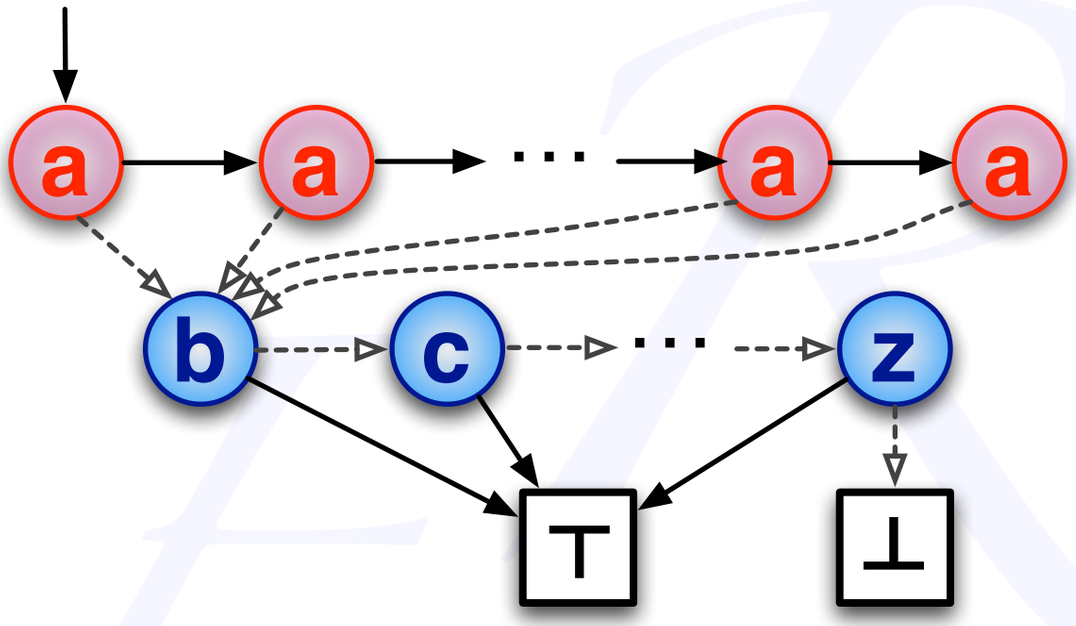
辺数：$(n+1)|\Sigma|-1$

漸近的に $|\Sigma|$ 倍

SequenceBDD における文字順序

$$L_n = \{\ \mathbf{a}^k\mathbf{x} \mid k \leqq n,\ \mathbf{x} \in \Sigma\backslash\{\mathbf{a}\}\ \}$$

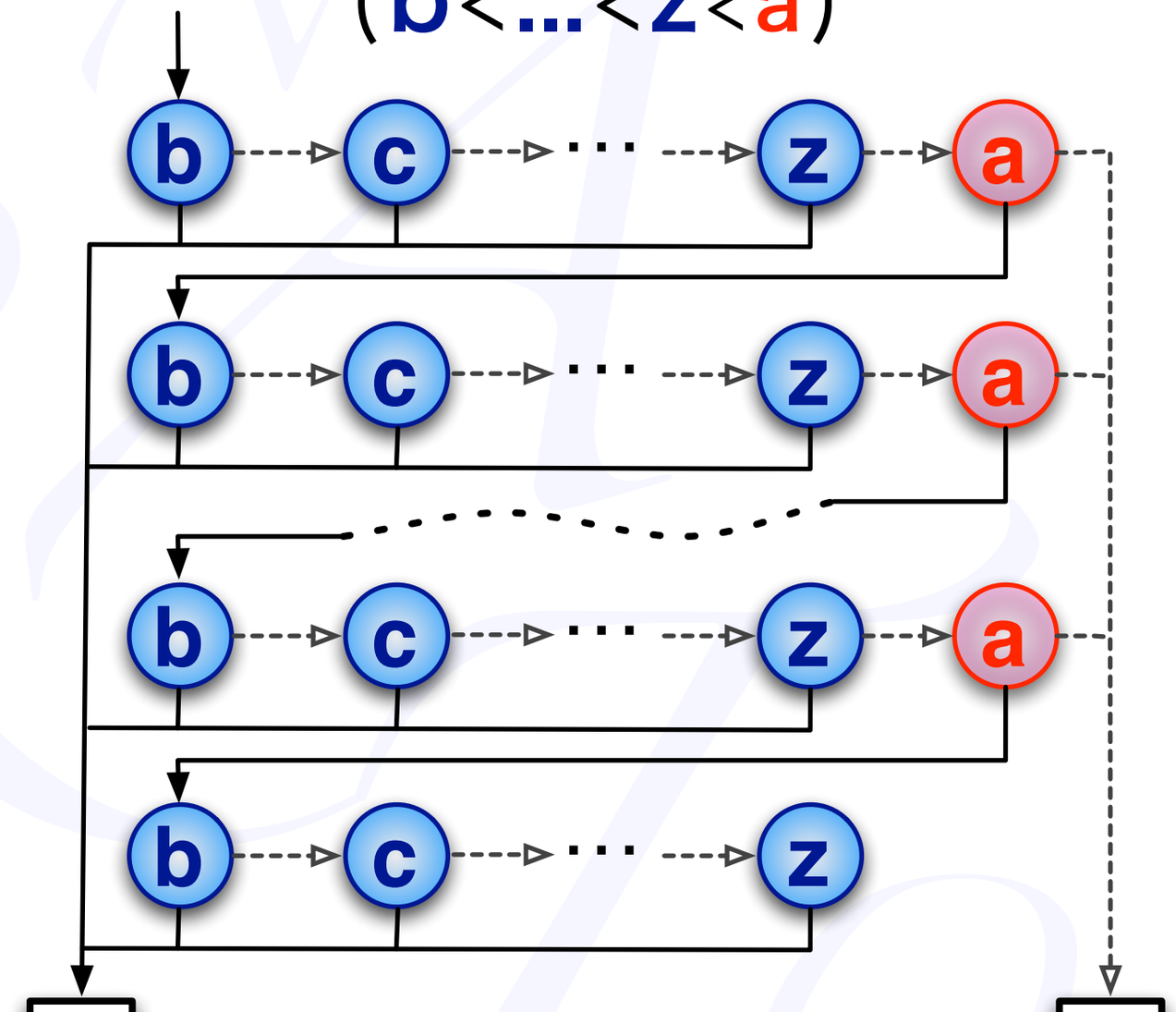
(a<b<...<z)



節点数：$n+|\Sigma|-1$

漸近的に $|\Sigma|$ 倍

(b<...<z<a)



節点数：$(n+1)|\Sigma|-1$

（minDFA の辺数に同じ）

Minimal DFA vs sequence BDD

- [minDFA 節点数] ≦ [seqBDD 節点数]
  ≦ [minDFA 辺数] ≦ [seqBDD 節点数]×|Σ|
  （ [seqBDD 節点数] は底節点数を数えてない）
- seq-BDD では文字順序の変更は高々
  |Σ| 倍程度の記述量変化しかもたらさない
  - ZDD では変数順序は指数的な記述量変化をもたらす
  - この２つの事実は矛盾しない

部分文字列
索引データ

文字列 w の全ての部分文字列を表現

- 全ての**接尾辞**を受理する DFA を作れば良い
  （ただし dead state 無し）
- 受理状態に至る　⇒　接尾辞
- 何らかの状態に至る　⇒　部分文字列
- 途中で遷移先が見つからない
  ⇒　部分文字列ではない

てゆーか受理状態とかいらないし

よく知られたDFAなデータ構造

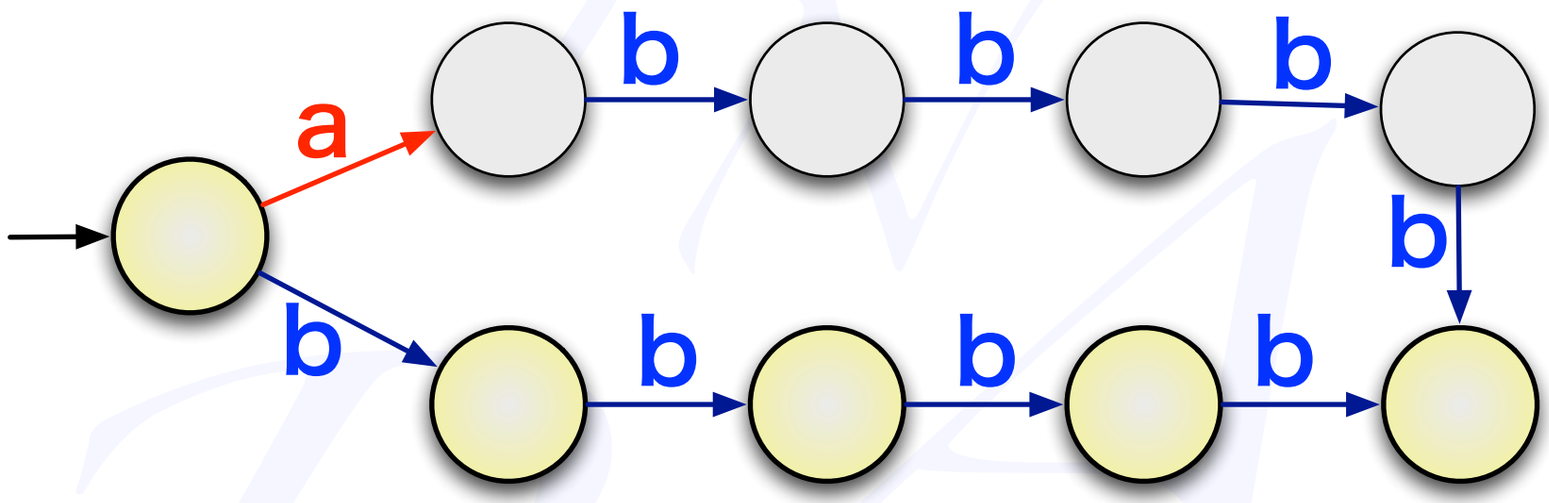
	目標	形状	サイズ
Suffix Trie	**接尾辞**	**木**	**$O(\|w\|^2)$**
DAWG (Suffix Automaton)	**接尾辞**	**DAG**	**$O(\|w\|)$**
Factor Automaton	**部分文字列**	**DAG**	**$O(\|w\|)$**

- Suffix Trie は最小ではない DFA といえる
- DAWG は Suffix Automaton から受理／拒否状態の区別をなくしたもの
- DAWG vs Factor Automaton
  - 一般には Factor Automaton が小さい
  - 最悪時の節点数・辺数に差はない
  - 入力サイズに対して線形時間のオンライン構築アルゴリズム

DAWG vs Factor Automaton

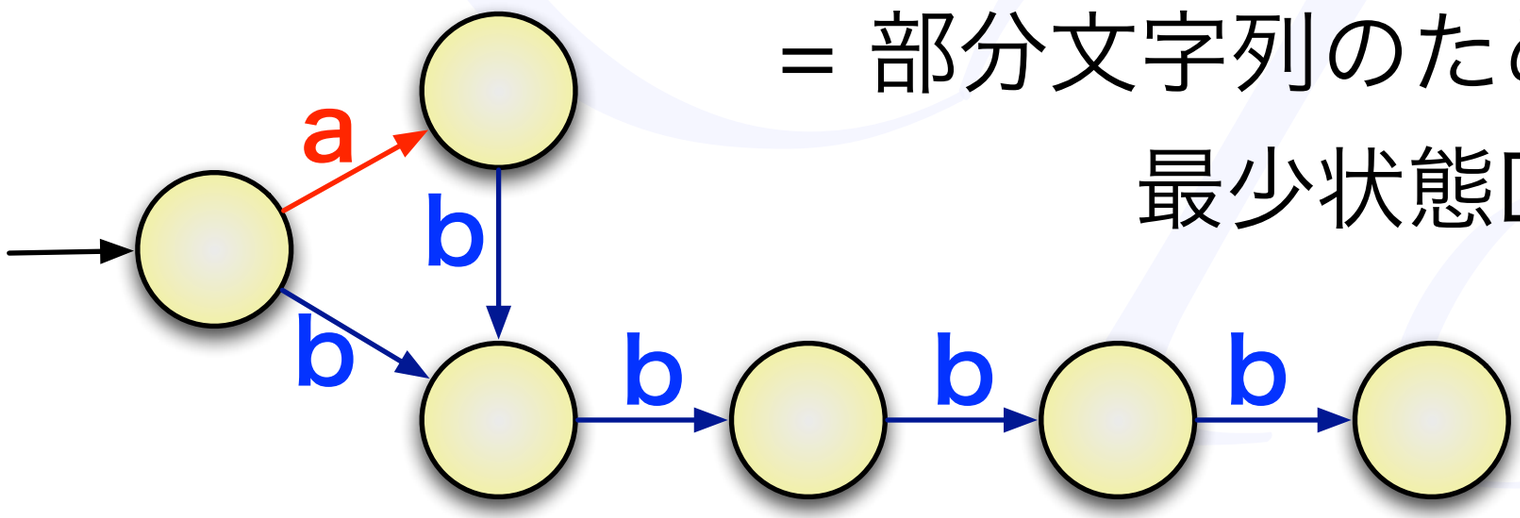
～　abbbb の場合　～

DAWG :



＊接尾辞たり得る場合とそうでない場合は節点を区別する

Factor Automaton :



＝ 部分文字列のための
最少状態DFA

Suffix-DD

- 伝住ら (DEIM'10)
- 文字列 w の全ての部分文字列を表現する seqBDD
  （DAWG よりも Factor Automaton に近い）
- Factor Automaton について知られている事実：
  - $|w| \leqq$ 辺数 $\leqq 3|w| - 4$
- よって Suffix-DD では
  - $|w| \leqq$ 節点数 $\leqq 3|w| - 4$
  - 右等号は $\mathbf{cb^n a}$ ($\mathbf{a<b<c}$) で成り立つ

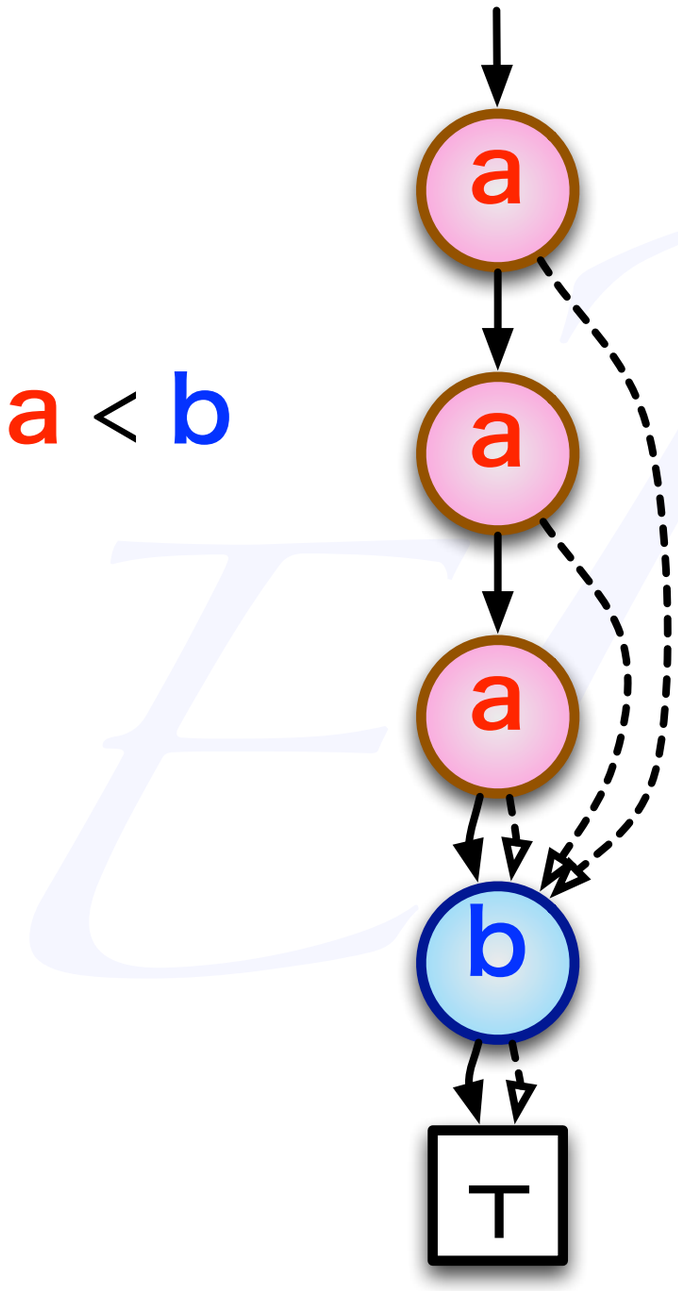
SuffixDD における文字順序

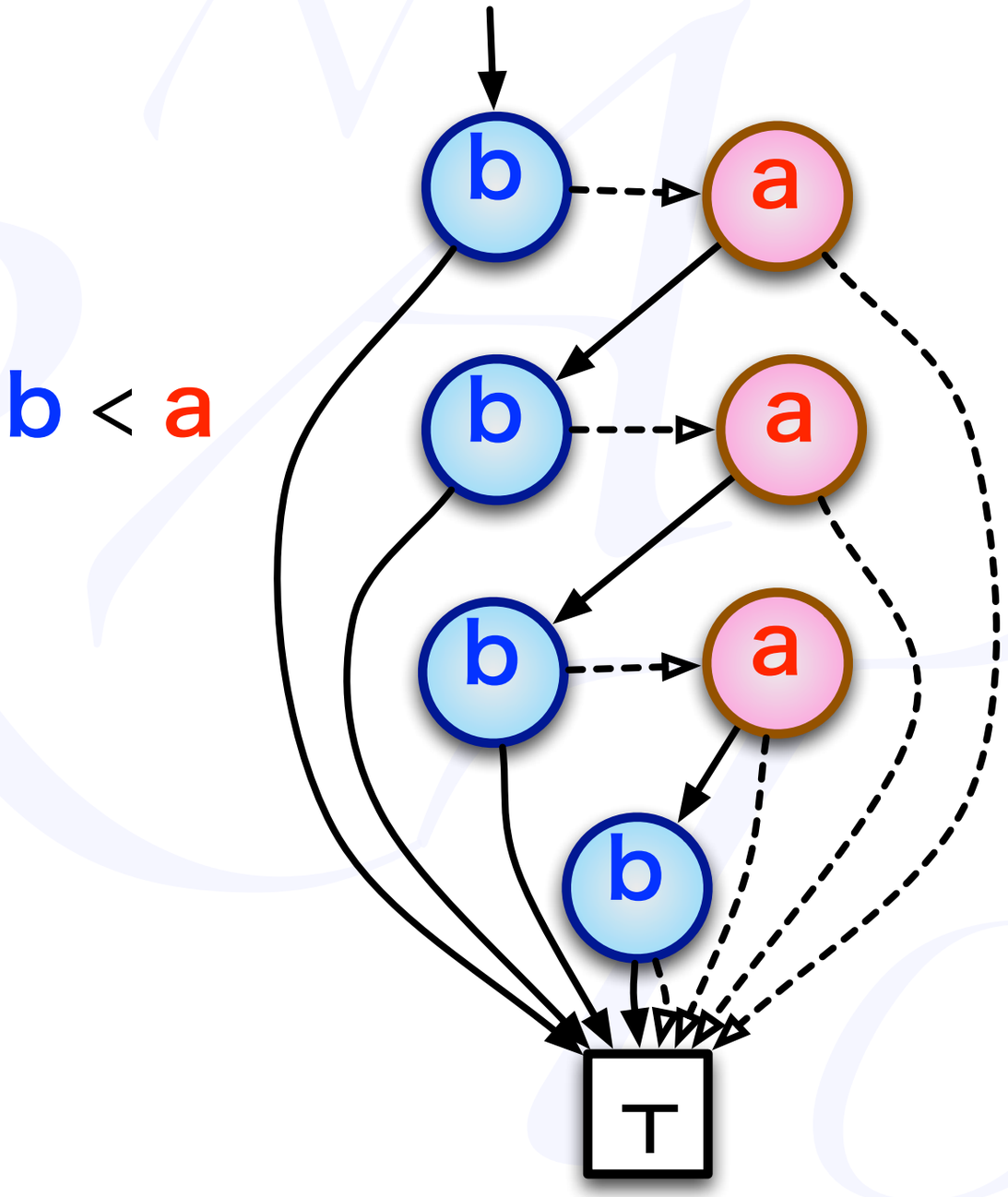
文字順序の変更　⇒　高々 |w|-1 の節点数変化

- Seq-BDD と minDFA について：
  - [seqBDDの節点数] ≦ [minDFAの辺数]
  - [minDFAの節点数] ≦ [seqBDDの節点数]+1
- Factor Automaton について知られている事実：
  - [FAの辺数] ≦ [FAの節点数] + |w| - 2
- よって
  - [seqBDDの節点数] ≦ [seqBDDの節点数] + |w| - 1

SuffixDD における文字順序

- 例：**aaab** の部分文字列を全てを表現





構築アルゴリズム

- Factor Automaton はオンライン線形時間で構築できる

構築アルゴリズム

- Factor Automaton はオンライン線形時間で構築できる

SuffixDD をオンライン線形時間で構築することは不可能

～　SeqBDD では各節点の表現する言語は不変　～

$w = a_1a_2...a_n$ の prefix $w_k = a_1a_2...a_k$ の部分文字列
$a_1...a_k$, $a_2...a_k$, ..., $a_{k-1}a_k$, a_k は,
$w_1, w_2, ..., w_{k-1}$ には見られなかった新しい文字列なので,
新しく k 個の節点を作らなければならない.

よって $\Omega(n^2)$ 時間かかる

主要参考文献

- SuffixTrie, SuffixTree の構築
  - Esko Ukkonen: “On-Line Construction of Suffix Trees”. *Algorithmica* **14**(3): 249-260 (1995)
- DAWG, Factor Automaton について
  - Maxime Crochemore: “Transducers and Repetitions”. *Theoretical Computer Science* **45**(1): 63-86 (1986)
  - A. Blumer, J. Blumer, D. Haussler, A. Ehrenfeucht, M. T. Chen, J. I. Seiferas “The Smallest Automaton Recognizing the Subwords of a Text”. *Theoretical Computer Science* **40**: 31-55 (1985)
- SequenceBDD について
  - E. Loekito, J. Bailey, and J. Pei, “A binary decision diagram based approach for mining frequent subsequences”. *Knowledge and Information Systems* **24**(2) 235-268, (2009)