



Title	SequenceBDDと既存のデータ構造
Author(s)	吉仲, 亮
Description	ERATO 세미나2010 : No.24. 2010年11月5日
Relation	2010年度科学技術振興機構ERATO湊離散構造処理系プロジェクト講究録. p.170-175.
Issue Date	2011-06
Doc URL	https://hdl.handle.net/2115/48461
Type	conference presentation
File Information	24_all.pdf



ERATO セミナ 2010 - No. 24

SequenceBDD と既存のデータ構造

吉仲 亮

JST ERATO 研究員

2010/11/5

概要

ZDD から派生したデータ構造として、文字列集合や系列 (sequential) データを表現するための seqBDD が提案されている。本発表では seqBDD の概説を行う。特に、1つの文字列の全ての部分文字列を表現する手法について、既存の DAWG などのデータ構造との比較を行う。

SequenceBDD と 既存のデータ構造

吉仲 亮
ERATOセミナー
2010/11/05

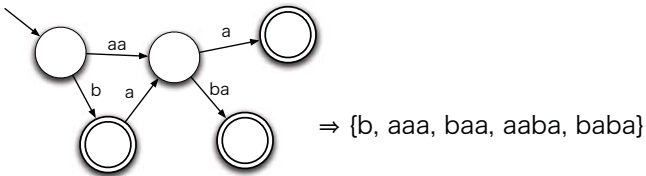
2010年11月10日水曜日

決定性有限状態機械 と sequenceBDD

2010年11月10日水曜日

決定性有限状態機械

- 文字列の(有限)集合の表現：
(無循環)部分決定性有限状態機械
(Acyclic Partial Deterministic Finite Automata)



2010年11月10日水曜日

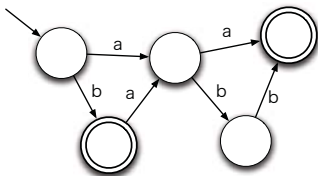
最少状態 DFA

- 正規言語に対して minDFA が一意に定まる
- 最小化 (Valmari & Lehtinen '08)
 - 時間： $O(|\text{辺の数}| \log |\text{節点数}|)$
 - 空間： $O(|\text{辺の数}|)$

2010年11月10日水曜日

DFA 上の演算

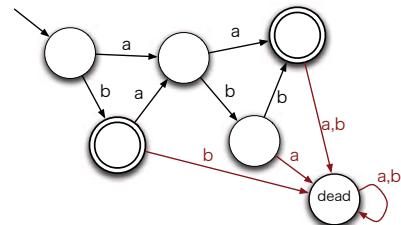
- 準備：partial ⇒ total
- dead state を作り、不足の辺は全て dead state へ



2010年11月10日水曜日

DFA 上の演算

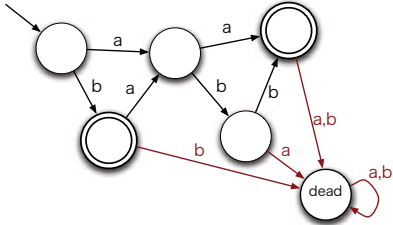
- 準備：partial ⇒ total
- dead state を作り、不足の辺は全て dead state へ



2010年11月10日水曜日

DFA 上の演算

- 準備: partial $\Rightarrow$ total
 - dead state を作り, 不足の辺は全て dead state へ



- **補集合**：受理状態と非受理状態の入れ替え

2010年11月10日水曜日

DFA 上の演算

- 2つのDFA M' , M'' 間の二項演算 (AND, OR etc.) :
直積DFA M を作り, 受理状態を適切に定める
 - $Q = Q' \times Q''$
-
- (q', q'') が受理 iff q' and/or q'' が受理
 - ただし、最小性は維持されない → あとから最小化



2010年11月10日水曜日

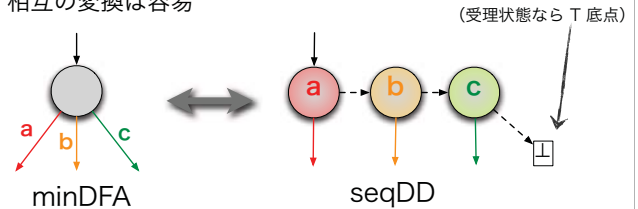
文字列集合上の演算

- 以上は教科書的な理論的な話
- 実際のDFAのための演算の実装アルゴリズム・・・？
- seq-BDD で文字列集合上の演算をする場合,
ZDD の実装の改変でだいぶ対応できる
- ただし, seq-BDD と非循環 DFA は理論上 “ ほぼ ” 同じ

2010年11月10日水曜日

Minimal DFA vs SequenceBDD

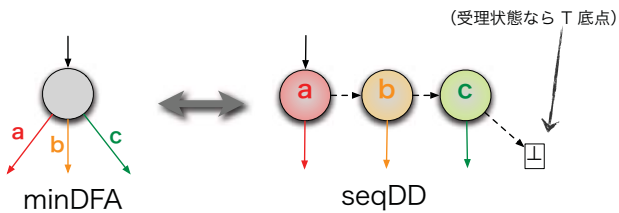
- 有限言語に対して一意な圧縮表現を与える
(seqBDD では文字順序固定のとき)
- minDFA の枝 $\leftrightarrow$ seqBDD の節点
- 相互の変換は容易



2010年11月10日水曜日

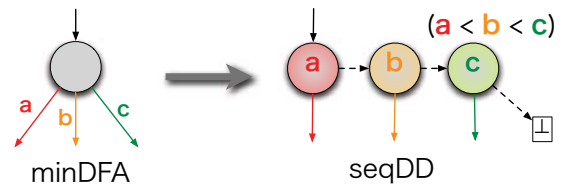
Minimal DFA vs SequenceBDD

- minDFA の枝 $\leftrightarrow$ 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



2010年11月10日水曜日

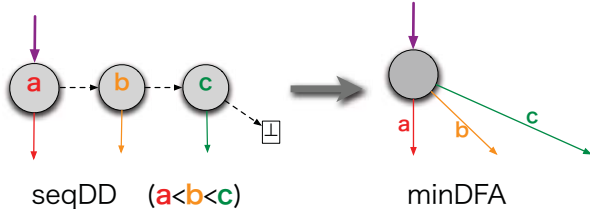
minDFA to seq-BDD



- ・ 非既約なseqDDへの変換（線形時間簡約化アルゴリズム）
- ・ $[\text{seqDD 節点数}] \leq [\text{minDFA 辺数}]$
- ・ minDFA のサイズは文字順序に依らないから、
文字順序最悪時の seq-BDD サイズと同程度には大きい

2010年11月10日水曜日

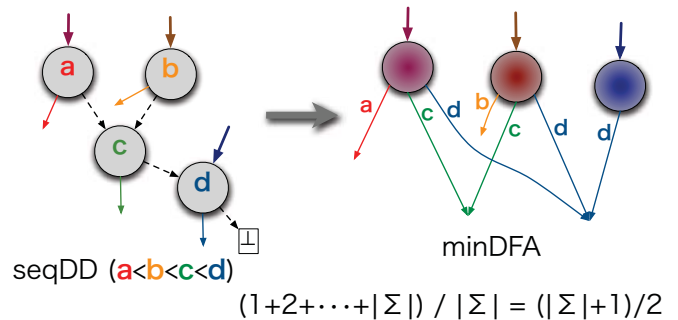
seq-BDD to minDFA



2010年11月10日水曜日

seq-BDD to minDFA

- $[\text{minDFA 辺数}] \leq [\text{seqDD 節点数}] \times (|\Sigma| + 1) / 2$



2010年11月10日水曜日

Minimal DFA vs sequential BDD

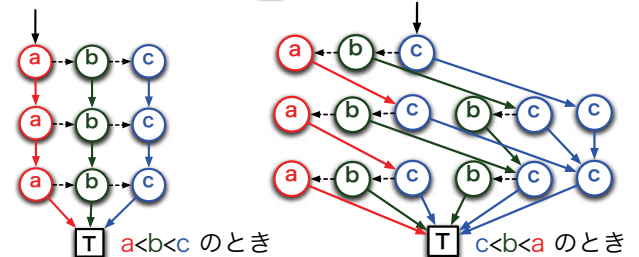
- $[\text{mDFA 節点数}] \leq [\text{sDD 節点数}]$
 $\leq [\text{mDFA 辺数}] \leq [\text{sDD 節点数}] \times (|\Sigma| + 1) / 2$
- ([sDD 節点数] は底節点数を数えてない)
- seq-BDD では文字順序の変更は高々 $|\Sigma|/2$ 程度の記述量変化しかもたらさない
- ZDD では変数順序は指数的な記述量変化をもたらす
- この2つの事実は矛盾しない

2010年11月10日水曜日

文字順序

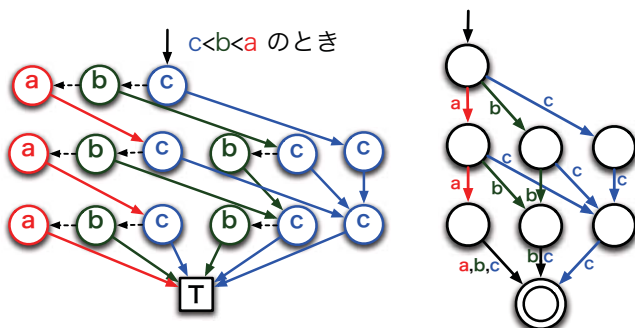
- seq-BDD では文字順序の変更の影響は高々 $|\Sigma|/2$ 倍.
- $\Sigma_n = \{ a_1, \dots, a_n \}$ に対して
 $L_{n,m} = \{ x_1 \dots x_m \mid x_i = a_j \text{ and } x_{i+1} = a_k \text{ implies } j \leq k \}$
 を格納する seq-BDD を考える

例: $L_{3,3}$ を表すseqDD ($\perp$ を指す枝を省略)



2010年11月10日水曜日

$L_{3,3}$ を受理する minDFA との比較



2010年11月10日水曜日

部分文字列データ

2010年11月10日水曜日

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 $\Rightarrow$ 接尾辞
- 何らかの状態に至る $\Rightarrow$ 部分文字列
- 途中で遷移先が見つからない $\Rightarrow$ 部分文字列ではない

2010年11月10日水曜日

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 $\Rightarrow$ 接尾辞
- 何らかの状態に至る $\Rightarrow$ 部分文字列
- 途中で遷移先が見つからない $\Rightarrow$ 部分文字列ではない

てゆーか受理状態とかいらんし

2010年11月10日水曜日

よく知られたDFAっぽいデータ構造

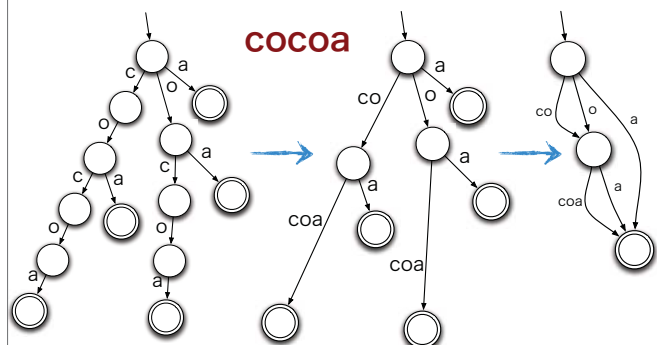
	目標	形状	辺ラベル	サイズ
Suffix Trie	接尾辞	木	1文字	$O(n^2)$
Suffix Tree	接尾辞	木	多数文字	$O(n)$
DAWG	接尾辞	DAG	1文字	$O(n)$
Factor Automaton	部分文字列	DAG	1文字	$O(n)$
CDAWG	接尾辞	DAG	多数文字	$O(n)$

- 多数文字：原文字列 w の部分文字列の位置によって多数文字ラベルを小空間で実現
- (C)DAWG = (Compact) Directed Acyclic Word Graph

いずれも出力サイズに対して線形時間のオンラインアルゴリズム

2010年11月10日水曜日

Suffix Trie, Suffix Tree, CDAWG



出次数1の節点は
明に書かない

等価な状態の統合

2010年11月10日水曜日

よく知られたDFAなデータ構造

	目標	形状	辺ラベル	サイズ
Suffix Trie	接尾辞	木	1文字	$O(n^2)$
Suffix Tree	接尾辞	木	多数文字	$O(n)$
DAWG	接尾辞	DAG	1文字	$O(n)$
Factor Automaton	部分文字列	DAG	1文字	$O(n)$
CDAWG	接尾辞	DAG	多数文字	$O(n)$

- 多数文字：原文字列 w の部分文字列の位置によって多数文字ラベルを小空間で実現

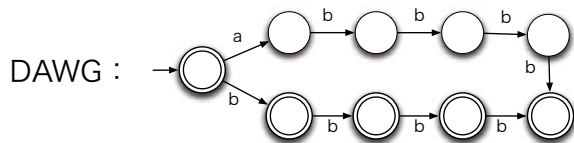
DAWG vs Factor Automaton

- 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

2010年11月10日水曜日

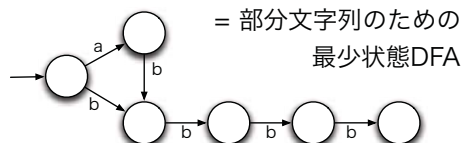
DAWG vs Factor Automaton

～ abbbb の場合 ～



* 接尾辞たり得る場合とそうでない場合は節点を統合しない

Factor Automaton :



2010年11月10日水曜日

Factor Automata

- $n+1 \leq \text{節点数} \leq 2n - 2$ (DAWG $\leq 2n-1$)
- $n \leq \text{辺の数} \leq 3n - 4$ (DAWG と同じ上限)
- オンライン線形時間アルゴリズム
- ごめんなさい難しくてわかっていません

2010年11月10日水曜日

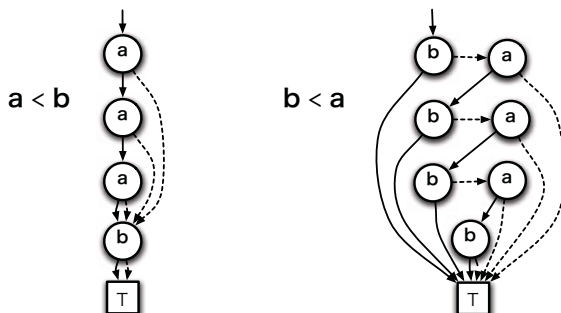
Suffix-DD

- 伝住ら (DEIM'10)
- 文字列 w の逆 w^R の全ての部分文字列を表現 (DAWG よりも Factor Automaton に近い)
- seqBDD の特殊な場合
 - $n+1 \leq \text{節点数} \leq 3n-2$ (Factor Automaton の辺数上限を与える例 (ab^nc) から)
 - よって文字列順序の影響もほとんどない

2010年11月10日水曜日

suffixDD における文字順序

- 部分文字列集合全体の格納に限ると文字順序は重要でない (Σ を固定しなくても定数倍にしかない)
- 2 倍になる例: aaab の部分文字列を全てを表現



2010年11月10日水曜日

Suffix-DD

- 文字列 w の逆 w^R の全ての部分文字列を表現
- 構築アルゴリズム
 - 逆部分文字列のオンライン平方時間 (伝住らによる)
 - 部分文字列のオフライン線形時間 (Factor Automaton 経由)
- FactorAutomaton のオンライン線形時間構築アルゴリズムを翻訳すれば、逆に部分文字列 seqBDD を構築できるはず

2010年11月10日水曜日

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, FactorAutomaton 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)

2010年11月10日水曜日