Model Checking for String Problems

Milka Hutagalung Martin Lange

School of Electr. Eng. and Comp. Sc., University of Kassel, Germany

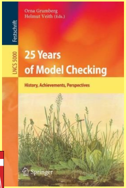
9th Int. Computer Science Symposium in Russia, CSR'14
09/06/2014

1. Model Checking
2. String Problems
3. The Polyadic μ-Calculus
4. An Algorithm for LCS

Model Checking – The Big Business

temporal logic

automata



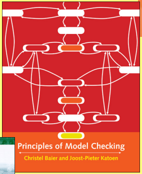
SAT/SMT

NuSMV

probabilistic systems

SPIN

real-time systems

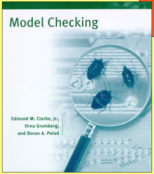


UPPAAL

quantitative m.c.

hybrid systems

BDDs



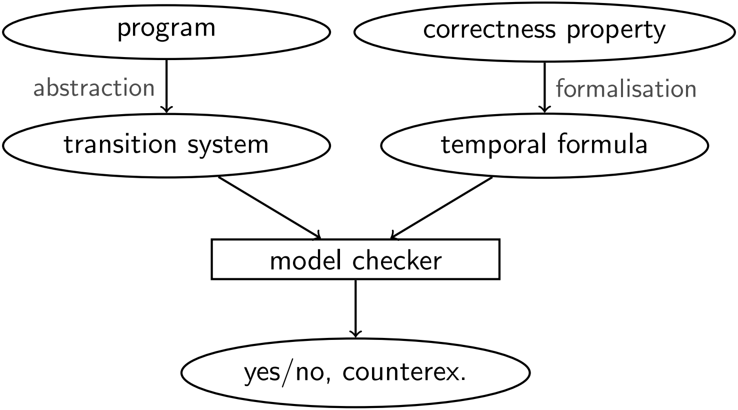
on-the-fly m.c.

fixpoints

bounded model checking

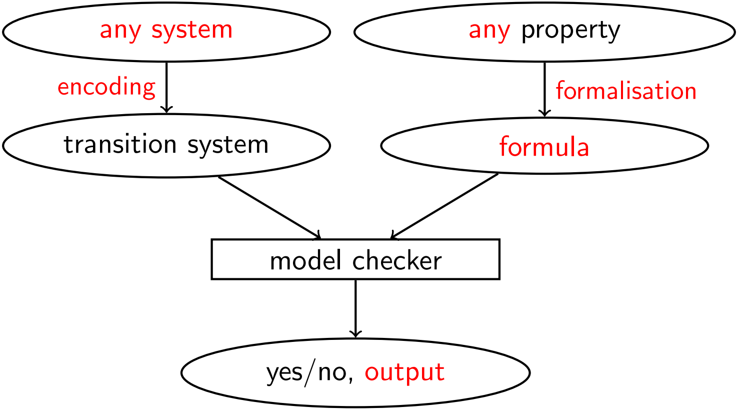
Model Checking – Be Generous

model checking as a method in program verification



Model Checking – Be Generous

model checking as a generic method for problem solving



Model Checking for Decision Problems

how to use model checking technology to solve decision problem P

1. take suitable logic $\mathcal{L}$ with decidable model checking problem
2. find suitable encoding enc for instances of P
3. find suitable defining formula φ_P

$$x \in P \quad \text{iff} \quad enc(x) \models \varphi_P$$

Model Checking for Decision Problems

how to use model checking technology to solve decision problem P

1. take suitable logic $\mathcal{L}$ with decidable model checking problem
2. find suitable encoding enc for instances of P
3. find suitable defining formula φ_P

$$x \in P \quad \text{iff} \quad enc(x) \models \varphi_P$$

4. optimise model checking algorithm for $\mathcal{L}$ into algorithm for P using partial evaluation

String Problems

input: $w_1, \ldots, w_n \in \Sigma^*$

compute

- longest common substring (LCST):
  v such that $w_i = \ldots v \ldots$ for all $i = 1, \ldots, n$

String Problems

input: $w_1, \ldots, w_n \in \Sigma^*$

compute

- longest common substring (LCST):
  v such that $w_i = \ldots v \ldots$ for all $i = 1, \ldots, n$
- shortest common superstring (SCST):
  v such that $v = \ldots w_i \ldots$ for all $i = 1, \ldots, n$

String Problems

input: $w_1, \dots, w_n \in \Sigma^*$

compute

- longest common substring (LCST):
  v such that $w_i = \dots v \dots$ for all $i = 1, \dots, n$
- shortest common superstring (SCST):
  v such that $v = \dots w_i \dots$ for all $i = 1, \dots, n$
- longest common subsequence (LCSS):
  $a_1 \dots a_k$ such that $w_i = \dots a_1 \dots a_2 \dots \dots a_k \dots$ for all $i = 1, \dots, n$

String Problems

input: $w_1, \ldots, w_n \in \Sigma^*$

compute

- longest common substring (LCST):
  v such that $w_i = \ldots v \ldots$ for all $i = 1, \ldots, n$
- shortest common superstring (SCST):
  v such that $v = \ldots w_i \ldots$ for all $i = 1, \ldots, n$
- longest common subsequence (LCSS):
  $a_1 \ldots a_k$ such that $w_i = \ldots a_1 \ldots a_2 \ldots \ldots a_k \ldots$ for all $i = 1, \ldots, n$

Algorithmic Solutions to String Problems

input: $w_1, \ldots, w_n \in \Sigma^*$, assuming $|w_i| \leq m$

- dynamic programming in $\mathcal{O}(m^n)$
- suffix tree algorithm in $\mathcal{O}(mn)$

String Problems via Computational Logic

goal: take suitable logic $\mathcal{L}$

define these string problems in $\mathcal{L}$ on suitable input representation

use model checking and partial evaluation to design algorithms for LCST, SCST, LCSS

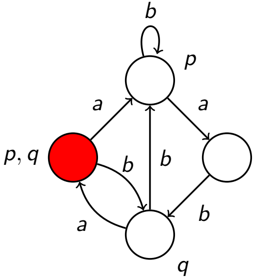
slight mismatch: LCST, SCST, LCSS are computation problems

The Modal μ-Calculus

syntax

$$\varphi ::= q \mid \overline{q} \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \langle a \rangle \varphi \mid [a]\varphi \mid X \mid \mu X.\varphi \mid \nu X.\varphi$$

interpreted in states of a transition system



$$\models \nu X.\mu Y.\langle b \rangle (q \wedge X) \vee \langle a \rangle Y$$

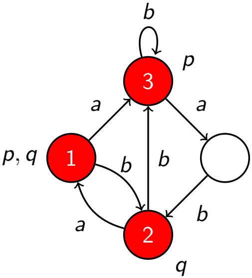
The Polyadic μ-Calculus

syntax [Andersen'94,Otto'99]

$$\varphi ::= q_i \mid \overline{q_i} \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \langle a \rangle_i \varphi \mid [a]_i \varphi \mid X \mid \mu X.\varphi \mid \nu X.\varphi \mid \{\bar{i} \leftarrow \bar{j}\}\varphi$$

with $1 \leq i \leq d$

interpreted in d-tuples of states of a transition system



$\models \langle b \rangle_1 \langle b \rangle_2 \langle a \rangle_3 (q_1 \wedge p_2)$

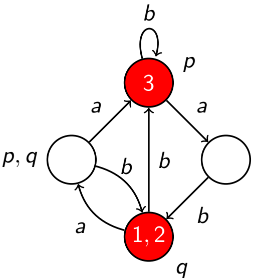
The μ-Calculus

syntax

$$\varphi ::= q_i \mid \overline{q_i} \mid \varphi\wedge\varphi \mid \varphi\vee\varphi \mid \langle a\rangle_i\varphi \mid [a]_i\varphi \mid X \mid \mu X.\varphi \mid \nu X.\varphi \mid \{\bar{i} \leftarrow \bar{j}\}\varphi$$

with $1 \leq i \leq d$

interpreted in d-tuples of states of a transition system



$\models \langle b\rangle_2\langle a\rangle_3(q_1 \wedge p_2)$

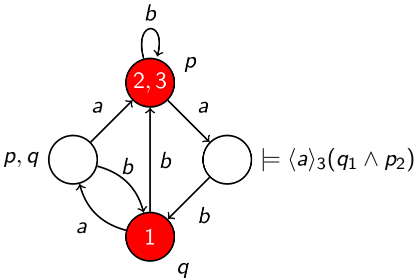
The μ-Calculus

syntax

$$\varphi ::= q_i \mid \overline{q_i} \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \langle a \rangle_i \varphi \mid [a]_i \varphi \mid X \mid \mu X.\varphi \mid \nu X.\varphi \mid \{\bar{i} \leftarrow \bar{j}\}\varphi$$

with $1 \leq i \leq d$

interpreted in d-tuples of states of a transition system



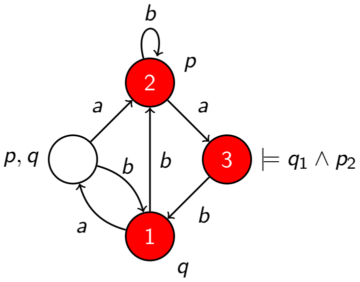
The μ-Calculus

syntax

$$\varphi ::= q_i \mid \overline{q_i} \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \langle a \rangle_i \varphi \mid [a]_i \varphi \mid X \mid \mu X.\varphi \mid \nu X.\varphi \mid \{\bar{i} \leftarrow \bar{j}\}\varphi$$

with $1 \leq i \leq d$

interpreted in d-tuples of states of a transition system



The Polyadic μ-Calculus

Fact: $\mathcal{L}_\mu = \mathcal{L}_\mu^1 \leq \mathcal{L}_\mu^2 \leq \ldots \leq \mathcal{L}_\mu^\omega$

standard example: $\mathcal{L}_\mu^2$ can define bisimilarity

$$\nu X.\big(\bigwedge_q \overline{q_1} \vee q_2\big) \wedge \big(\bigwedge_a [a]_1 \langle a \rangle_2 X\big) \wedge \{(1,2) \leftarrow (2,1)\}X$$

other simulations definable [Andersen'94, L./Lozes/Vargas'12]

The Polyadic μ-Calculus

Fact: $\mathcal{L}_\mu = \mathcal{L}_\mu^1 \le \mathcal{L}_\mu^2 \le \ldots \le \mathcal{L}_\mu^\omega$

standard example: $\mathcal{L}_\mu^2$ can define bisimilarity

$$\nu X.\big(\bigwedge_q \overline{q_1} \vee q_2\big) \wedge \big(\bigwedge_a [a]_1 \langle a \rangle_2 X\big) \wedge \{(1,2) \leftarrow (2,1)\} X$$

other simulations definable [Andersen'94, L./Lozes/Vargas'12]

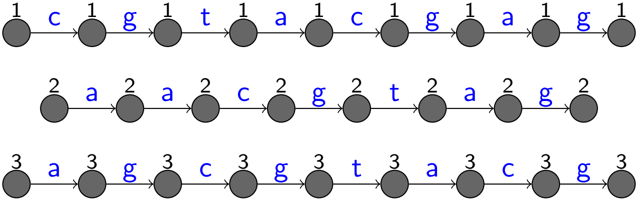
Thm.: $\mathcal{L}_\mu^\omega = \mathrm{PTIME}/{\sim}$ [Otto'99]

Thm.: $\mathcal{L}_\mu^2$ satisfiability is undecidable [Otto'99]

Thm.: $\mathcal{L}_\mu^d$ model checking in $\mathcal{O}((en^k)^d)$ [L./Lozes'12]

Defining String Problems in $\mathcal{L}_\mu^\omega$

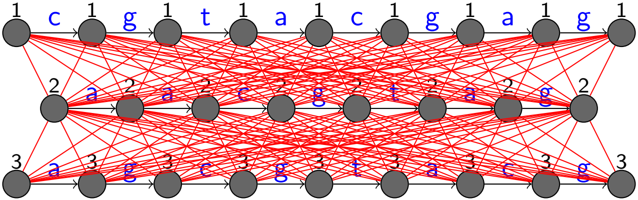
consider input $w_1, \ldots, w_n$ as transition system, e.g.



$$\Phi_{\mathrm{LCST}} = \nu X.\Big(\bigwedge_{i=1} i_i\Big) \wedge \bigvee_{b \in \{\mathrm{a,c,g,t}\}} \langle b\rangle_1 \langle b\rangle_2 \langle b\rangle_3 X$$

Defining String Problems in $\mathcal{L}_\mu^\omega$

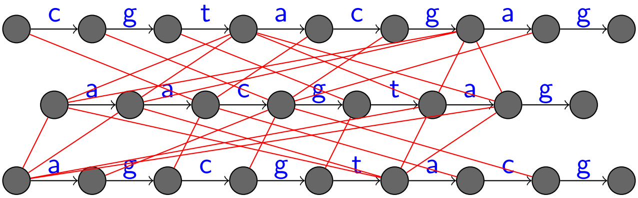
consider input $w_1, \dots, w_n$ as transition system, e.g.



$$\Phi_{\mathsf{LCST}} = \nu X.\big(\bigwedge_{i=1} i_i\big) \wedge \bigvee_{b\in\{\mathrm{a,c,g,t}\}} \langle b\rangle_1\langle b\rangle_2\langle b\rangle_3 X$$

Defining String Problems in $\mathcal{L}_\mu^\omega$

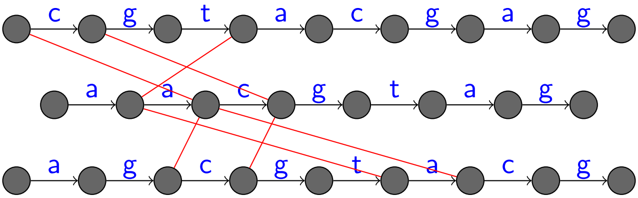
consider input $w_1, \ldots, w_n$ as transition system, e.g.



$$\Phi_{\mathrm{LCST}} = \nu^1 X.\big(\bigwedge_{i=1} i_i\big) \wedge \bigvee_{b\in\{\mathrm{a,c,g,t}\}} \langle b\rangle_1 \langle b\rangle_2 \langle b\rangle_3 X$$

Defining String Problems in $\mathcal{L}_\mu^\omega$

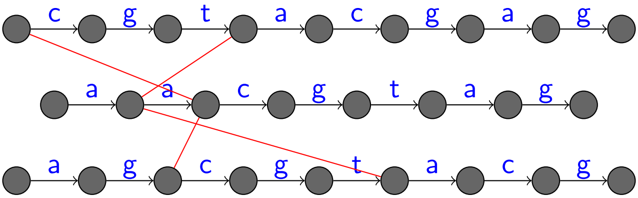
consider input $w_1, \ldots, w_n$ as transition system, e.g.



$$\Phi_{\mathrm{LCST}} = \nu^{2} X.\Big(\bigwedge_{i=1} i_i \Big) \wedge \bigvee_{b \in \{\mathrm{a,c,g,t}\}} \langle b \rangle_1 \langle b \rangle_2 \langle b \rangle_3 X$$

Defining String Problems in $\mathcal{L}_{\mu}^{\omega}$

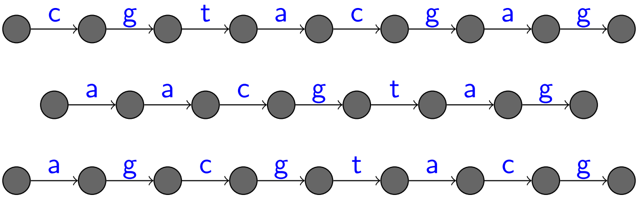
consider input $w_1, \ldots, w_n$ as transition system, e.g.



$$\Phi_{\mathsf{LCST}} = \nu^{3} X.\big(\bigwedge_{i=1} i_i \big) \wedge \bigvee_{b \in \{\mathsf{a},\mathsf{c},\mathsf{g},\mathsf{t}\}} \langle b \rangle_1 \langle b \rangle_2 \langle b \rangle_3 X$$

Defining String Problems in $\mathcal{L}_\mu^\omega$

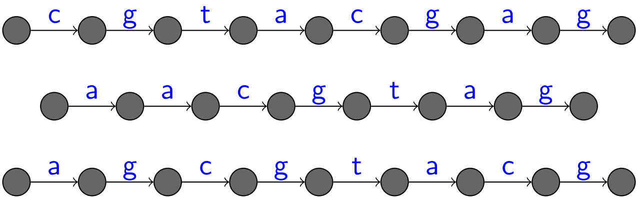
consider input $w_1, \ldots, w_n$ as transition system, e.g.



$$\Phi_{\mathsf{LCST}} = \nu^{4} X.\Big(\bigwedge_{i=1} i_i\Big) \wedge \bigvee_{b \in \{\mathsf{a},\mathsf{c},\mathsf{g},\mathsf{t}\}} \langle b\rangle_1 \langle b\rangle_2 \langle b\rangle_3 X$$

Defining String Problems in $\mathcal{L}_\mu^\omega$

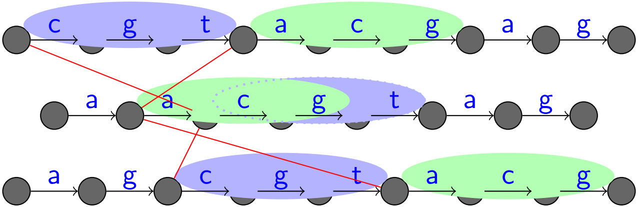
consider input $w_1, \ldots, w_n$ as transition system, e.g.



$$\Phi_{\mathsf{LCST}} = \nu X.\big(\bigwedge_{i=1} i_i\big) \wedge \bigvee_{b\in\{\mathsf{a,c,g,t}\}} \langle b\rangle_1\langle b\rangle_2\langle b\rangle_3 X \quad \equiv \quad \bot$$

Defining String Problems in $\mathcal{L}_\mu^\omega$

consider input $w_1, \ldots, w_n$ as transition system, e.g.



$$\Phi_{\mathrm{LCST}} = \nu^{3} X.\big(\bigwedge_{i=1} i_i \big) \wedge \bigvee_{b \in \{\mathrm{a,c,g,t}\}} \langle b \rangle_1 \langle b \rangle_2 \langle b \rangle_3 X$$

longest common substrings acg and cgt found in penultimate iteration

Using Partial Evaluation

observations on model checking Φ_{LCST}

- deterministic transition relation
- sets of n-tuples with exactly one state per input word
- operations: union, left-shift-by-one, intersection
- needed in each step: occurrences of $\mathrm{a}, \mathrm{c}, \mathrm{g}, \mathrm{t}$

maintain occurrence sets for $b \in \{\mathrm{a}, \mathrm{c}, \mathrm{g}, \mathrm{t}\}$

$$Occ(b) \ := \ \{j_i \mid w_i(j) = b\}$$

Using Partial Evaluation

observations on model checking Φ_{LCST}

- deterministic transition relation
- sets of n-tuples with exactly one state per input word
- operations: union, left-shift-by-one, intersection
- needed in each step: occurrences of $\mathrm{a}, \mathrm{c}, \mathrm{g}, \mathrm{t}$

maintain occurrence sets for $b \in \{\mathrm{a}, \mathrm{c}, \mathrm{g}, \mathrm{t}\}$

$$Occ(b) \;:=\; \{j_i \mid w_i(j) = b\}$$

extend to occurrences of subwords $w \in \{\mathrm{a}, \mathrm{c}, \mathrm{g}, \mathrm{t}\}^+$ by

$$Occ(bw) \;:=\; Occ(b) \cap \{j_i \mid (j+1)_i \in Occ(w)\}$$

Solutions

Def.: w is solution if $\forall i\ \exists j.\ j_i \in Occ(w)$

Lemma: bw is solution $\Rightarrow$ w is solution

Solutions

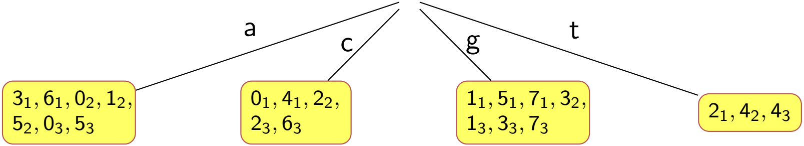
Def.: w is solution if $\forall i\ \exists j.\ j_i \in Occ(w)$

Lemma: bw is solution $\Rightarrow$ w is solution

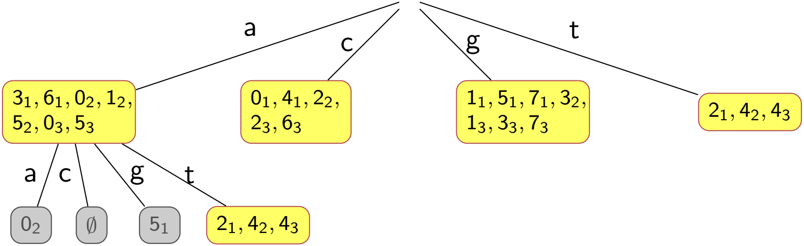
Algorithm LCST

procedure $\mathrm{LCST}(w_1, \ldots, w_n)$
  $\mathcal{S} \leftarrow \Sigma$
  compute $Occ(b)$ for all $b \in \Sigma$
  repeat
    $\mathcal{S}' \leftarrow \mathcal{S}$
    $\mathcal{S} \leftarrow \{bw \mid w \in \mathcal{S}', b \in \Sigma\}$
    compute $Occ(bw)$ for all $bw \in \mathcal{S}$
    eliminate non-solutions from $\mathcal{S}$
  until $\mathcal{S} = \emptyset$
  return $\{(w, Occ(w)) \mid w \in \mathcal{S}'\}$
end procedure

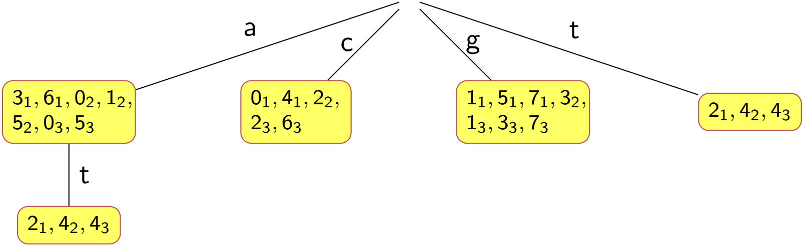
Example (Continued)



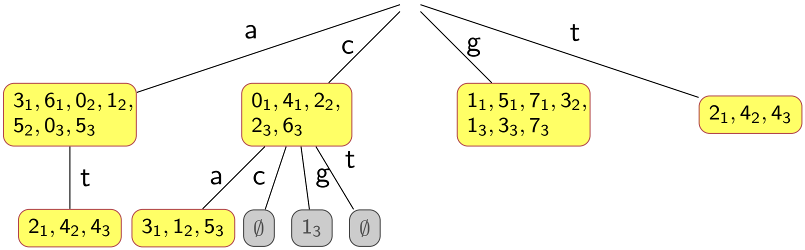
Example (Continued)



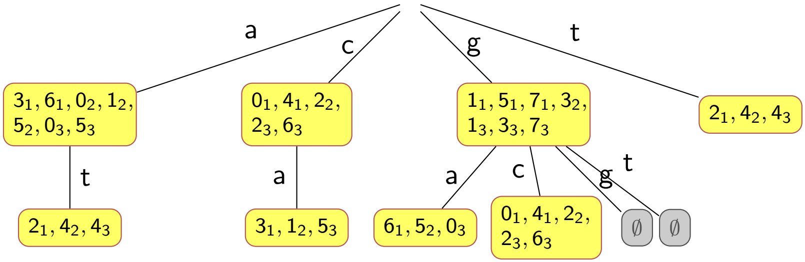
Example (Continued)



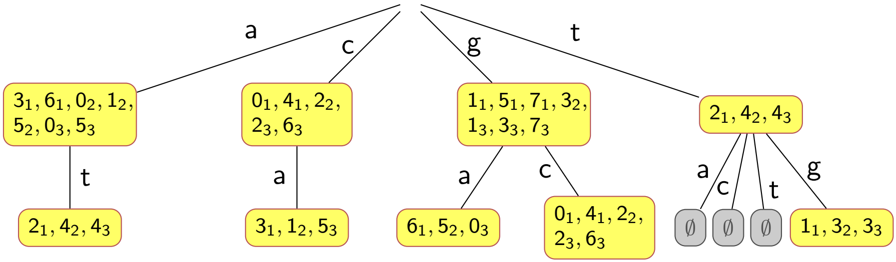
Example (Continued)



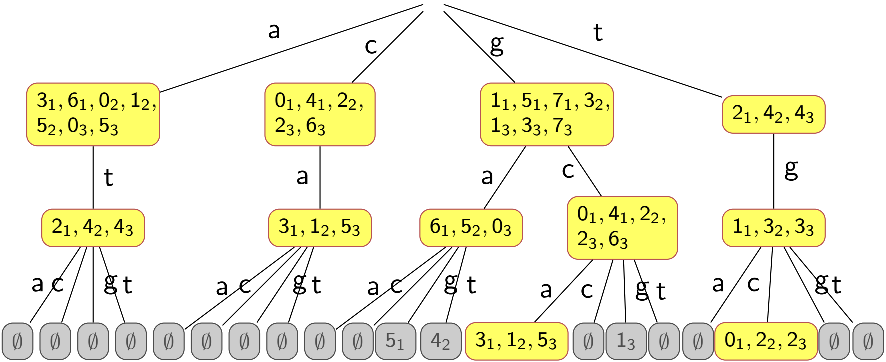
Example (Continued)



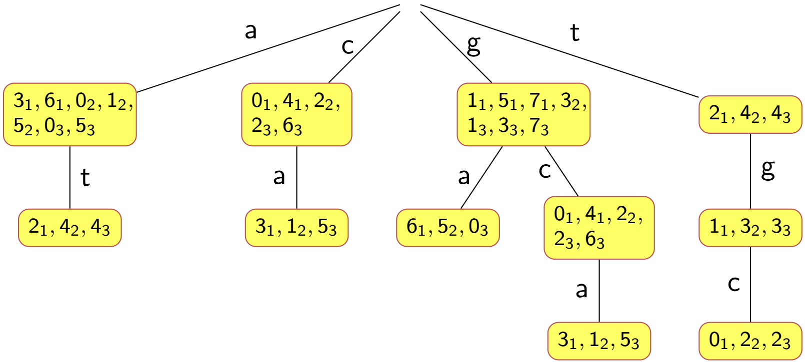
Example (Continued)



Example (Continued)



Example (Continued)



Further Optimisations

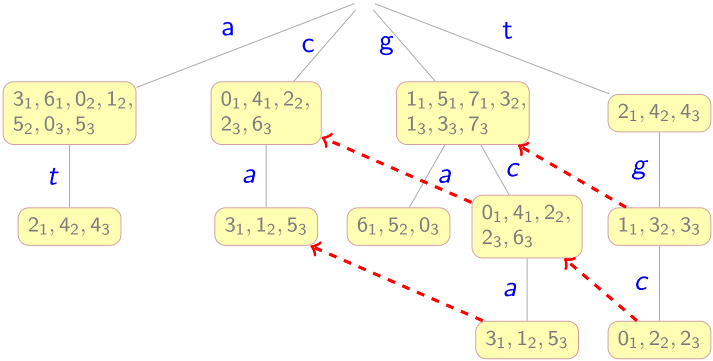
for instance: bw not a solution $\Rightarrow$ bwb' not a solution

$\leadsto$ point to longest prefix

Further Optimisations

for instance: bw not a solution $\Rightarrow$ bwb' not a solution

$\rightsquigarrow$ point to longest prefix



Complexity Analysis

input $w_1, \ldots, w_n$ with $|w_i| \leq m$

Theorem 1

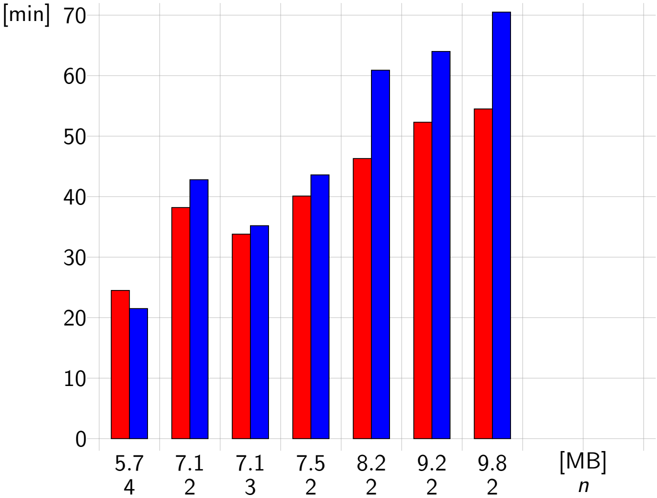
Algorithm LCST *can be made to explore at most* $\mathcal{O}(nm)$ *many occurrence sets.*

PROOF IDEA: partition nodes in this graph into three kinds

- non-extendable to the left $\leq m$
- extendable to the left by multiple letters $\leq nm$
- others $\leq m$

Benchmark

LCST against suffix tree algorithm on genome analysis between bacteria / viri



Online Potential

conceptual comparison between LCST and suffix tree algorithm

- similar tree-like data structure $\rightsquigarrow$ similar optimisations possible
- built differently
  - suffix tree alg. processes input words one-by-one
  - LCST processes common subwords one-by-one

  rule of thumb: after half the running time
  - suffix tree alg. finds longest common substring of half the inputs
  - LCST finds half-longest common substring of all inputs

The End