

Hyper-minimisation of weighted finite automata

Daniel Quernheim and Andreas Maletti



Institute for Natural Language Processing, University of Stuttgart
 <first.last>@ims.uni-stuttgart.de

August 18, 2011

Outline

Unweighted case

Weighted case

Conclusion

Outline

Unweighted case

Weighted case

Conclusion

Minimisation

Problem

given DFA, return

- ▶ equivalent DFA such that
- ▶ all equivalent DFA are larger

Theorem (Hopcroft 1971)

DFA minimisation can be done in time $O(n \log n)$

- ▶ *n : number of states*

Minimisation

Problem

given DFA, return

- ▶ equivalent DFA
- ▶ minimal

Theorem (Hopcroft 1971)

DFA minimisation can be done in time $O(n \log n)$

- ▶ *n : number of states*

Hyper-minimisation

Definition

DFA A, B **almost equivalent** if $L(A)$ and $L(B)$ have finite difference

Problem [Badr et al. 2009]

given DFA, return

- ▶ **almost** equivalent DFA such that
- ▶ all **almost** equivalent DFA are larger

Theorem (Holzer, ~ 2009, Gawrychowsky, Jež 2009)

DFA hyper-minimisation can be done in time $O(n \log n)$

Hyper-minimisation

Definition

DFA A, B **almost equivalent** if $L(A)$ and $L(B)$ have finite difference

Problem [Badr et al. 2009]

given DFA, return

- ▶ **almost** equivalent DFA
- ▶ hyper-minimal

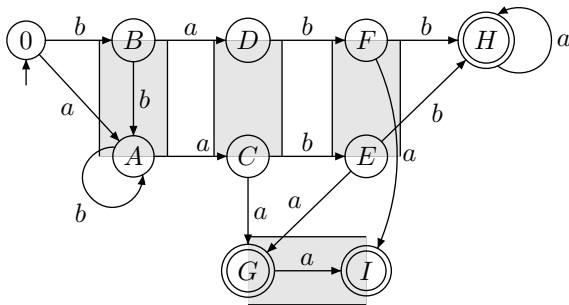
Theorem (Holzer, ~ 2009, Gawrychowsky, Jež 2009)

DFA hyper-minimisation can be done in time $O(n \log n)$

Almost equivalence

- States are **almost equivalent** if their right languages differ finitely

$$\vec{q} = \{w \in \Sigma^* \mid \delta(q, w) \in F\}$$

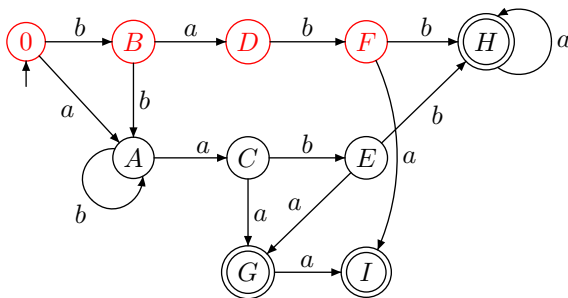


Kernel and preamble states

Definition

- ▶ **preamble state**: finite left language
- ▶ **kernel state**: infinite left language

$$\overleftarrow{q} = \{w \in \Sigma^* \mid \delta(q_0, w) = q\}$$



Structural characterisation

Theorem (Badr et al. 2009)

DFA is hyper-minimal iff

- ▶ *minimal*
- ▶ *no preamble state is almost equivalent to another state*

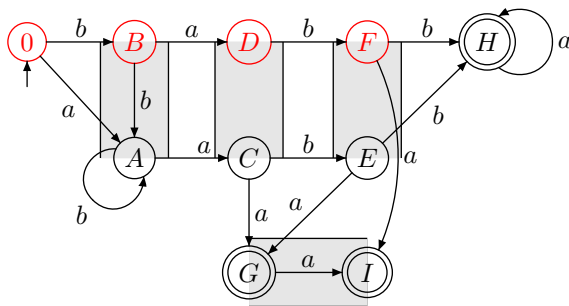
Structural characterisation

Theorem (Badr et al. 2009)

DFA is hyper-minimal iff

- ▶ *minimal*
- ▶ *no preamble state is almost equivalent to another state*

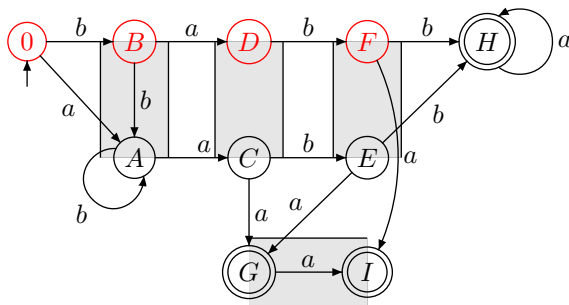
Example (Not hyper-minimal)



Hyper-minimisation

preamble	almost equivalent
<i>B</i>	<i>A</i>
<i>D</i>	<i>C</i>
<i>F</i>	<i>E</i>

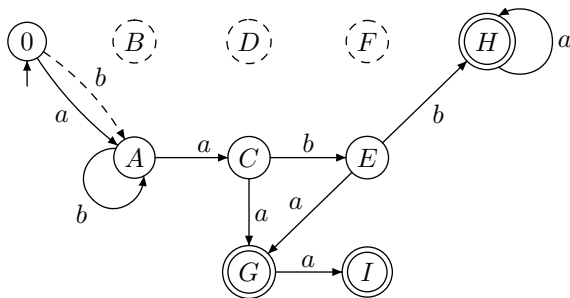
Table: Merges



Hyper-minimisation

preamble	almost equivalent
B	A
D	C
F	E

Table: Merges



Motivation

Application

In natural language processing

- ▶ Automata often deterministic for efficient evaluation
- ▶ DFA tend to be very big → good reduction potential
- ▶ Often approximative data → lossy compression ok

Motivation

Application

In natural language processing

- ▶ Automata often deterministic for efficient evaluation
- ▶ DFA tend to be very big → good reduction potential
- ▶ Often approximative data → lossy compression ok
- ▶ **but DFA are weighted**

Conclusion

Let's do hyper-minimisation for weighted DFA!

Outline

Unweighted case

Weighted case

Conclusion

Weight structure

General approach

works for **semifields** (semirings with mult. inverses)

Weight structure

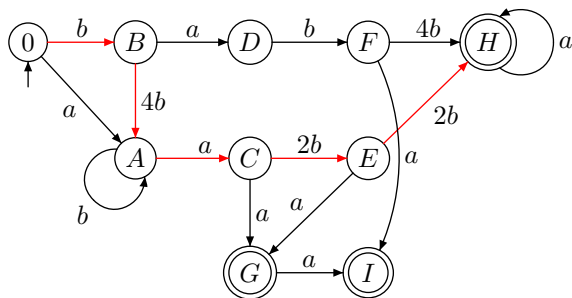
General approach

works for **semifields** (semirings with mult. inverses)

Presentation

here we use the field of real numbers

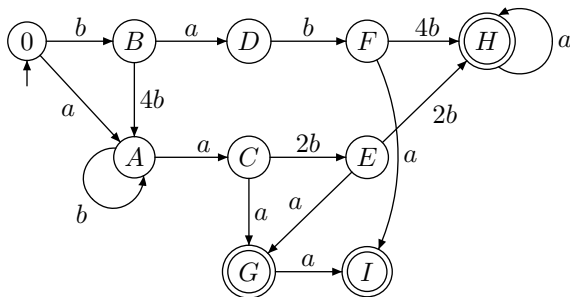
Example (W DFA)



weight of $bbabb$
is $1 \cdot 4 \cdot 1 \cdot 2 \cdot 2 = 16$

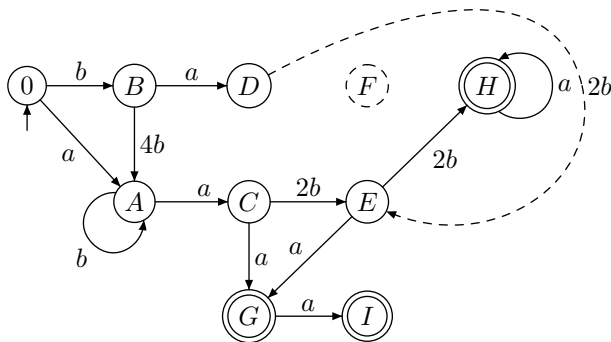
weight of ab is 0

Weighted merge



Weighted merge of F into E with factor 2

Weighted merge



Weighted merge of F into E with factor 2

Almost equivalence

Definition

Two WDFA A, B are **almost equivalent** if

$$A(w) \neq B(w) \quad \text{for only finitely many } w \in \Sigma^*$$

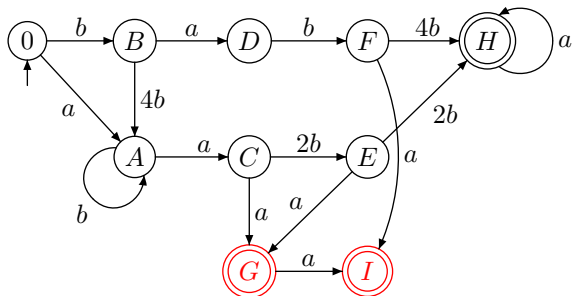
Definition

Two states p, q are **almost equivalent** if there is $k \in \mathbb{R} \setminus \{0\}$ such that

$$\vec{p}(w) \neq k \cdot \vec{q}(w) \quad \text{for only finitely many } w \in \Sigma^*$$

[Borchardt: The Myhill-Nerode theorem for recognizable tree series. DLT 2003]

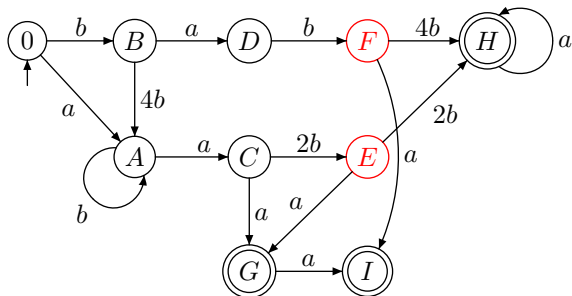
Example



	<i>G</i>	<i>I</i>
ϵ	1	1
a	1	0

► *G* and *I* almost equivalent

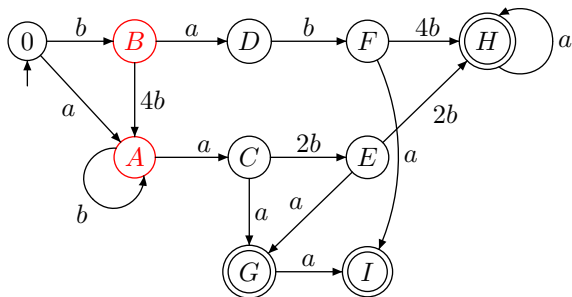
Example



	<i>E</i>	<i>F</i>
<i>a</i>	1	1
<i>aa</i>	1	0
<i>b</i>	2	4
<i>ba</i>	2	4
<i>baa</i>	2	4
<i>baaa</i>	2	4
...		

- ▶ *G* and *I* almost equivalent
- ▶ *E* and *F* almost equivalent (with factor 2)

Example



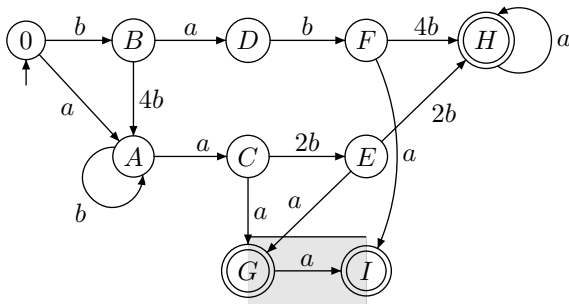
	<i>A</i>	<i>B</i>
<i>abb</i>	4	4
<i>abba</i>	4	4
<i>abbaa</i>	4	4
...		
<i>babb</i>	4	16
<i>babba</i>	4	16
<i>babbaa</i>	4	16
...		

- ▶ *G* and *I* almost equivalent
- ▶ *E* and *F* almost equivalent (with factor 2)
- ▶ *A* and *B* **not almost equivalent!**

Finding almost equivalent states

Definition

- ▶ **co-kernel state**: infinite right language
- ▶ **co-preamble state**: finite right language



Signature standardisation

Definition

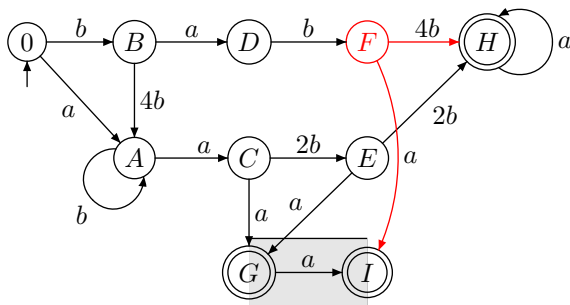
signature of q : $(\langle \delta(q, \sigma), c(q, \sigma) \rangle)_{\sigma \in \Sigma}$

Standardisation

- ▶ select minimal $\sigma_0 \in \Sigma$ such that $\delta(q, \sigma_0)$ is co-kernel
- ▶ adjust transition weights:

$$c'(q, \sigma) = \begin{cases} \frac{c(q, \sigma)}{c(q, \sigma_0)} & \text{if } \delta(q, \sigma) \text{ is co-kernel} \\ 1 & \text{otherwise} \end{cases}$$

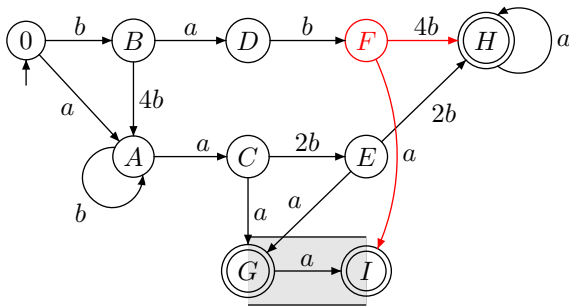
Signature standardisation



Standardised signature of F

- $F \xrightarrow{a} I$ leads to co-preamble

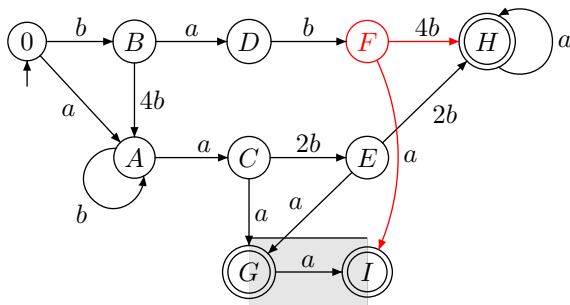
Signature standardisation



Standardised signature of F

- $F \xrightarrow{a} I$ leads to co-preamble $\rightarrow c'(F, a) = 1$

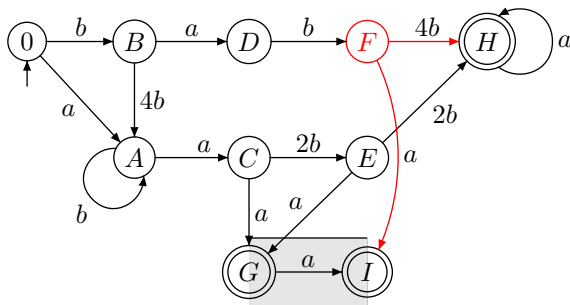
Signature standardisation



Standardised signature of F

- ▶ $F \xrightarrow{a} I$ leads to co-preamble $\rightarrow c'(F, a) = 1$
- ▶ $F \xrightarrow{4b} H$ leads to co-kernel

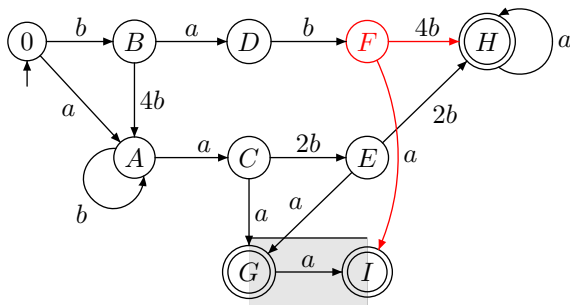
Signature standardisation



Standardised signature of F

- ▶ $F \xrightarrow{a} I$ leads to co-preamble $\rightarrow c'(F, a) = 1$
- ▶ $F \xrightarrow{4b} H$ leads to co-kernel $\rightarrow c'(F, b) = \frac{4}{4} = 1$

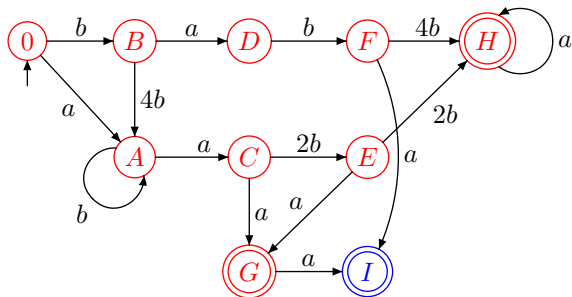
Signature standardisation



Standardised signature of F

- ▶ $F \xrightarrow{a} I$ leads to co-preamble $\rightarrow c'(F, a) = 1$
- ▶ $F \xrightarrow{4b} H$ leads to co-kernel $\rightarrow c'(F, b) = \frac{4}{4} = 1$
- ▶ **Standardised signature $(\langle I, 1 \rangle, \langle H, 1 \rangle)$**

Finding almost equivalent states



Signature map:

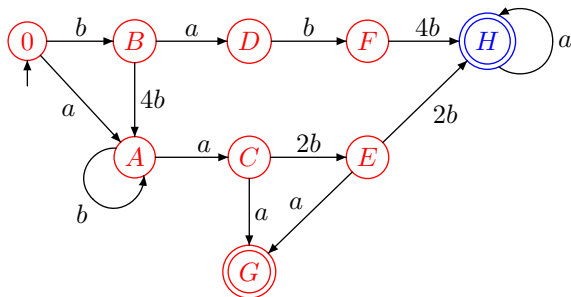
$\langle \perp, 1 \rangle, \langle \perp, 1 \rangle$	$\perp$
--	---------

Blocks:

- $\{\perp, I\}$

- $\text{sig}(I) = (\langle \perp, 1 \rangle, \langle \perp, 1 \rangle)$ in map!
- add I to block of $\perp$ (and merge)

Finding almost equivalent states



Signature map:

$\langle \perp, 1 \rangle, \langle \perp, 1 \rangle$	$\perp$
$\langle H, 1 \rangle, \langle \perp, 1 \rangle$	H

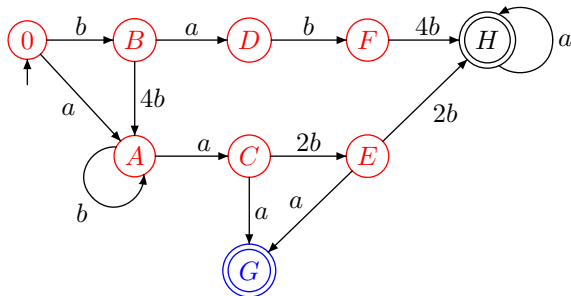
Blocks:

- $\{\perp, I\}$

► $\text{sig}(H) = (\langle H, 1 \rangle, \langle \perp, 1 \rangle)$

► add to map

Finding almost equivalent states



Signature map:

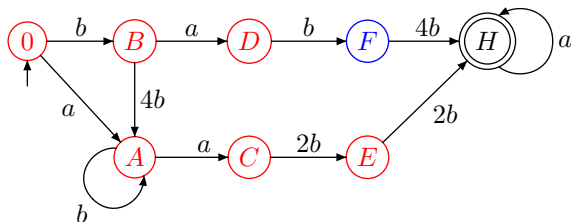
$\langle \perp, 1 \rangle, \langle \perp, 1 \rangle$	$\perp$
$\langle H, 1 \rangle, \langle \perp, 1 \rangle$	H

Blocks:

► $\{\perp, I, G\}$

- $\text{sig}(G) = (\langle \perp, 1 \rangle, \langle \perp, 1 \rangle)$ in map!
- add G to $\perp$ (and merge)

Finding almost equivalent states



Signature map:

$\langle \perp, 1 \rangle, \langle \perp, 1 \rangle$	$\perp$
$\langle H, 1 \rangle, \langle \perp, 1 \rangle$	H
$\langle \perp, 1 \rangle, \langle H, 1 \rangle$	F

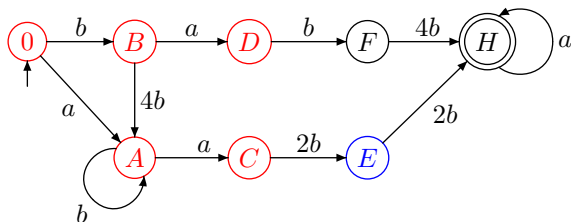
Blocks:

► $\{\perp, I, G\}$

► $\text{sig}(F) = (\langle \perp, 1 \rangle, \langle H, 1 \rangle)$

► add to map

Finding almost equivalent states



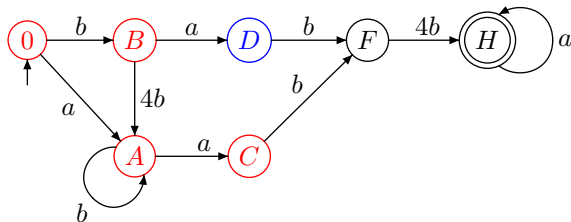
Signature map:

$\langle \perp, 1 \rangle, \langle \perp, 1 \rangle$	$\perp$
$\langle H, 1 \rangle, \langle \perp, 1 \rangle$	H
$\langle \perp, 1 \rangle, \langle H, 1 \rangle$	F

Blocks:

- ▶ $\{\perp, I, G\}$
- ▶ $\{F, E\}$
- ▶ $\text{sig}(E) = (\langle \perp, 1 \rangle, \langle H, 1 \rangle)$ in map!
- ▶ add E to F (and merge with factor $\frac{1}{2}$)

Finding almost equivalent states



Signature map:

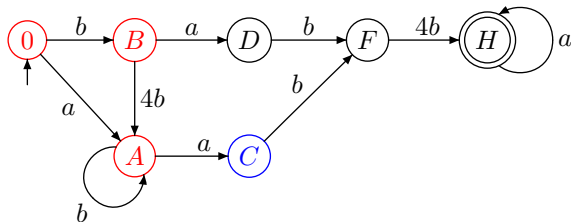
$\langle \perp, 1 \rangle, \langle \perp, 1 \rangle$	$\perp$
$\langle H, 1 \rangle, \langle \perp, 1 \rangle$	H
$\langle \perp, 1 \rangle, \langle H, 1 \rangle$	F
$\langle \perp, 1 \rangle, \langle F, 1 \rangle$	D

Blocks:

- ▶ $\text{sig}(D) = (\langle \perp, 1 \rangle, \langle F, 1 \rangle)$
- ▶ add to map

- ▶ $\{\perp, I, G\}$
- ▶ $\{F, E\}$

Finding almost equivalent states



Signature map:

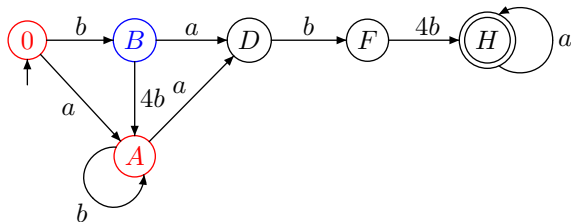
$\langle \perp, 1 \rangle, \langle \perp, 1 \rangle$	$\perp$
$\langle H, 1 \rangle, \langle \perp, 1 \rangle$	H
$\langle \perp, 1 \rangle, \langle H, 1 \rangle$	F
$\langle \perp, 1 \rangle, \langle F, 1 \rangle$	D

Blocks:

- ▶ $\text{sig}(C) = (\langle \perp, 1 \rangle, \langle F, 1 \rangle)$ in map!
- ▶ Add C to D (and merge)

- ▶ $\{\perp, I, G\}$
- ▶ $\{F, E\}$
- ▶ $\{D, C\}$

Finding almost equivalent states



- ▶ $\text{sig}(B) = (\langle D, 1 \rangle, \langle A, 4 \rangle)$
- ▶ add to map

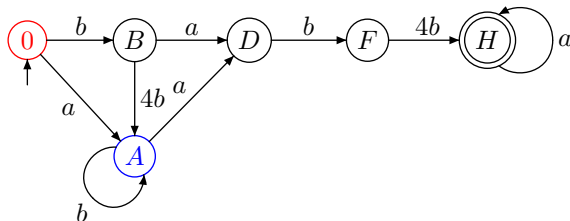
Signature map:

$\langle \perp, 1 \rangle, \langle \perp, 1 \rangle$	$\perp$
$\langle H, 1 \rangle, \langle \perp, 1 \rangle$	H
$\langle \perp, 1 \rangle, \langle H, 1 \rangle$	F
$\langle \perp, 1 \rangle, \langle F, 1 \rangle$	D
$\langle D, 1 \rangle, \langle A, 4 \rangle$	B

Blocks:

- ▶ $\{\perp, I, G\}$
- ▶ $\{F, E\}$
- ▶ $\{D, C\}$

Finding almost equivalent states



- ▶ $\text{sig}(A) = (\langle D, 1 \rangle, \langle A, 1 \rangle)$
- ▶ add to map

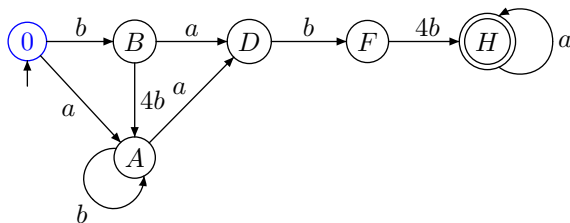
Signature map:

$\langle \perp, 1 \rangle, \langle \perp, 1 \rangle$	$\perp$
$\langle H, 1 \rangle, \langle \perp, 1 \rangle$	H
$\langle \perp, 1 \rangle, \langle H, 1 \rangle$	F
$\langle \perp, 1 \rangle, \langle F, 1 \rangle$	D
$\langle D, 1 \rangle, \langle A, 4 \rangle$	B
$\langle D, 1 \rangle, \langle A, 1 \rangle$	A

Blocks:

- ▶ $\{\perp, I, G\}$
- ▶ $\{F, E\}$
- ▶ $\{D, C\}$

Finding almost equivalent states



- ▶ $\text{sig}(0) = (\langle A, 1 \rangle, \langle B, 1 \rangle)$
- ▶ add to map

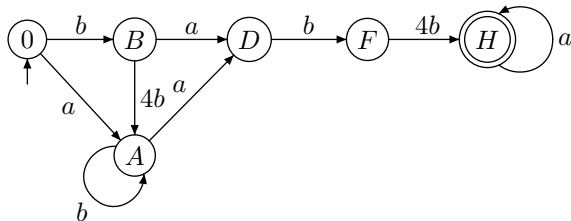
Signature map:

$\langle \perp, 1 \rangle, \langle \perp, 1 \rangle$	$\perp$
$\langle H, 1 \rangle, \langle \perp, 1 \rangle$	H
$\langle \perp, 1 \rangle, \langle H, 1 \rangle$	F
$\langle \perp, 1 \rangle, \langle F, 1 \rangle$	D
$\langle D, 1 \rangle, \langle A, 4 \rangle$	B
$\langle D, 1 \rangle, \langle A, 1 \rangle$	A
$\langle A, 1 \rangle, \langle B, 1 \rangle$	0

Blocks:

- ▶ $\{\perp, I, G\}$
- ▶ $\{F, E\}$
- ▶ $\{D, C\}$

Finding almost equivalent states



Signature map:

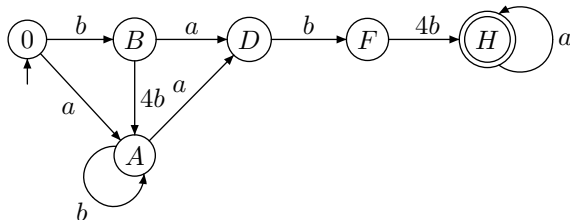
$\langle \perp, 1 \rangle, \langle \perp, 1 \rangle$	$\perp$
$\langle H, 1 \rangle, \langle \perp, 1 \rangle$	H
$\langle \perp, 1 \rangle, \langle H, 1 \rangle$	F
$\langle \perp, 1 \rangle, \langle F, 1 \rangle$	D
$\langle D, 1 \rangle, \langle A, 4 \rangle$	B
$\langle D, 1 \rangle, \langle A, 1 \rangle$	A
$\langle A, 1 \rangle, \langle B, 1 \rangle$	0

Blocks:

- ▶ Blocks represent almost equivalence
- ▶ Scaling factors used in merges

- ▶ $\{\perp, I, G\}$
- ▶ $\{F, E\}$
- ▶ $\{D, C\}$

Finding almost equivalent states



Signature map:

$\langle \perp, 1 \rangle, \langle \perp, 1 \rangle$	$\perp$
$\langle H, 1 \rangle, \langle \perp, 1 \rangle$	H
$\langle \perp, 1 \rangle, \langle H, 1 \rangle$	F
$\langle \perp, 1 \rangle, \langle F, 1 \rangle$	D
$\langle D, 1 \rangle, \langle A, 4 \rangle$	B
$\langle D, 1 \rangle, \langle A, 1 \rangle$	A
$\langle A, 1 \rangle, \langle B, 1 \rangle$	0

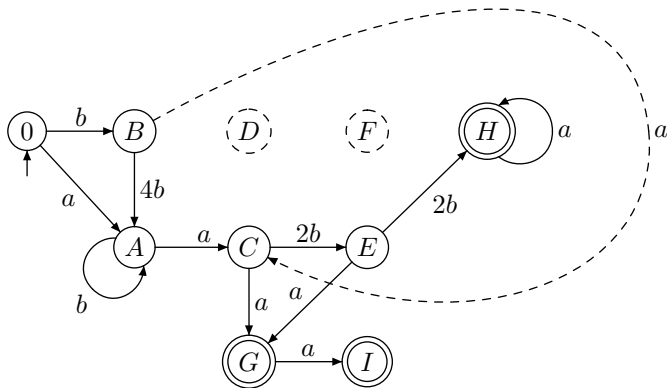
Blocks:

- ▶ Blocks represent almost equivalence
- ▶ Scaling factors used in merges
- ▶ **but we merged kernel states**

- ▶ $\{\perp, I, G\}$
- ▶ $\{F, E\}$
- ▶ $\{D, C\}$

Weighted merges

- ▶ merge of F into E with factor 2
- ▶ merge of D into C with factor 1



Conclusion

Solved

- ▶ hyper-minimisation for WDFA over semifields

Open

- ▶ Optimise number of errors
- ▶ Incremental construction
- ▶ Extensions to WNFA, more weight structures, etc.

Conclusion

Solved

- ▶ hyper-minimisation for WDFA over semifields

Open

- ▶ Optimise number of errors
- ▶ Incremental construction
- ▶ Extensions to WNFA, more weight structures, etc.

Thank you for your attention!

References



Andrew Badr, Viliam Geffert, and Ian Shipman.

Hyper-minimizing minimized deterministic finite state automata.

RAIRO Theor. Inf. Appl., 43(1), 2009.



Jason Eisner.

Simpler and more general minimization for weighted finite-state automata.

In *Proc. HLT-NAACL*, 2003.



Paweł Gawrychowski and Artur Jeż.

Hyper-minimisation made efficient.

In *Proc. MFCS*, volume 5734 of *LNCS*, 2009.



Markus Holzer and Andreas Maletti.

An $n \log n$ algorithm for hyper-minimizing a (minimized) deterministic automaton.

Theor. Comput. Sci., 411(38–39), 2010.



John E. Hopcroft.

An $n \log n$ Algorithm for Minimizing the States in a Finite Automaton.

In *The Theory of Machines and Computations*. Academic Press, 1971.