Deciding language equivalence for unambiguous VASSes

Wojciech Czerwiński

Piotr Hofman

Plan

Plan

- basic notions

Plan

- basic notions
- equivalence for deterministic VASSes

Plan

- basic notions
- equivalence for deterministic VASSes
- regular-separability for VASSes

Plan

- basic notions
- equivalence for deterministic VASSes
- regular-separability for VASSes
- future-determinisation

Plan

- basic notions
- equivalence for deterministic VASSes
- regular-separability for VASSes
- future-determinisation
- further work

Language of a VASS

Language of a VASS

VASS = vector addition system with states

Language of a VASS

VASS = vector addition system with states

configuration = (state, vector in N^d)

Language of a VASS

VASS = vector addition system with states

configuration = (state, vector in $\mathbb{N}^d$)

$$p(u) \xrightarrow[(p,v,q)]{a} q(u+v)$$

Language of a VASS

VASS = vector addition system with states

configuration = (state, vector in N^d)

$$p(u) \xrightarrow[(p,v,q)]{a} q(u+v)$$

initial configuration, acceptance by states

Language of a VASS

VASS = vector addition system with states

configuration = (state, vector in $\mathbb{N}^d$)

$$p(u) \xrightarrow[(p,v,q)]{a} q(u+v)$$

initial configuration, acceptance by states

detVASS = each reachable configuration is deterministic

Equivalence for detVASS

Equivalence for detVASS

Ackermann-hard (reduction from reachability)

Equivalence for detVASS

Ackermann-hard (reduction from reachability)

Inclusion is decidable

Equivalence for detVASS

Ackermann-hard (reduction from reachability)

Inclusion is decidable

Witness for non-inclusion:

Equivalence for detVASS

Ackermann-hard (reduction from reachability)

Inclusion is decidable

Witness for non-inclusion:

1) word from (in_1, in_2) to (acc_1, rej_2)

Equivalence for detVASS

Ackermann-hard (reduction from reachability)

Inclusion is decidable

Witness for non-inclusion:

1) word from (in_1, in_2) to (acc_1, rej_2)

2) word from (in_1, in_2) to (c_1, c_2) where c_1 can fire a and reach acceptance, c_2 is a-blocking

Equivalence for detVASS

Ackermann-hard (reduction from reachability)

Inclusion is decidable

Witness for non-inclusion:

1) word from (in_1, in_2) to (acc_1, rej_2)

2) word from (in_1, in_2) to (c_1, c_2) where c_1 can fire a and reach acceptance, c_2 is a-blocking

Reachability of (upward closed, downward closed)

Reachability of (U, D)

Reachability of (U, D)

Reachability of (upward closed, downward closed)

Reachability of (U, D)

Reachability of (upward closed, downward closed)

upward closed = finite union of p(u)↑

Reachability of (U, D)

Reachability of (upward closed, downward closed)

upward closed = finite union of $p(u)\uparrow$

downward closed = finite union of $q(v)\downarrow$

Reachability of (U, D)

Reachability of (upward closed, downward closed)

upward closed = finite union of $p(u)\uparrow$

downward closed = finite union of $q(v)\downarrow$

$v = (3, \omega, 5, \omega)$

Reachability of (U, D)

Reachability of (upward closed, downward closed)

upward closed = finite union of $p(u)\uparrow$

downward closed = finite union of $q(v)\downarrow$

$v = (3, \omega, 5, \omega)$

Add auxiliary transitions and check reachability of (u, v')

Reachability of (U, D)

Reachability of (upward closed, downward closed)

upward closed = finite union of $p(u)\uparrow$

downward closed = finite union of $q(v)\downarrow$

$v = (3, \omega, 5, \omega)$

Add auxiliary transitions and check reachability of (u,v')

Decidable also for extended VASSes
(downward closed set prohibited)

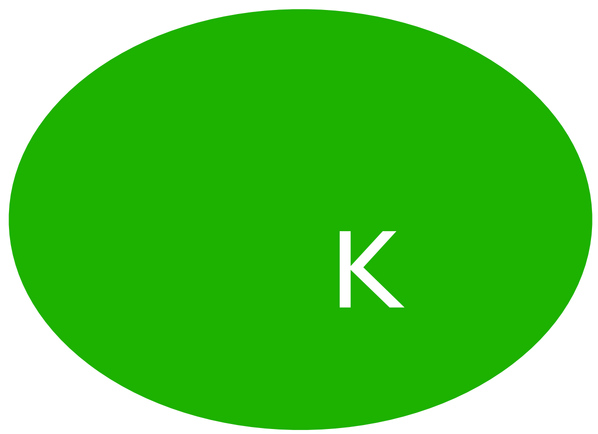
Regular-separability

Regular-separability

Languages K and L are regular-separable
if there is a regular S such that
$K \subseteq S$ and $L \cap S = \varnothing$.

Regular-separability

Languages K and L are regular-separable
if there is a regular S such that
$K \subseteq S$ and $L \cap S = \varnothing$.

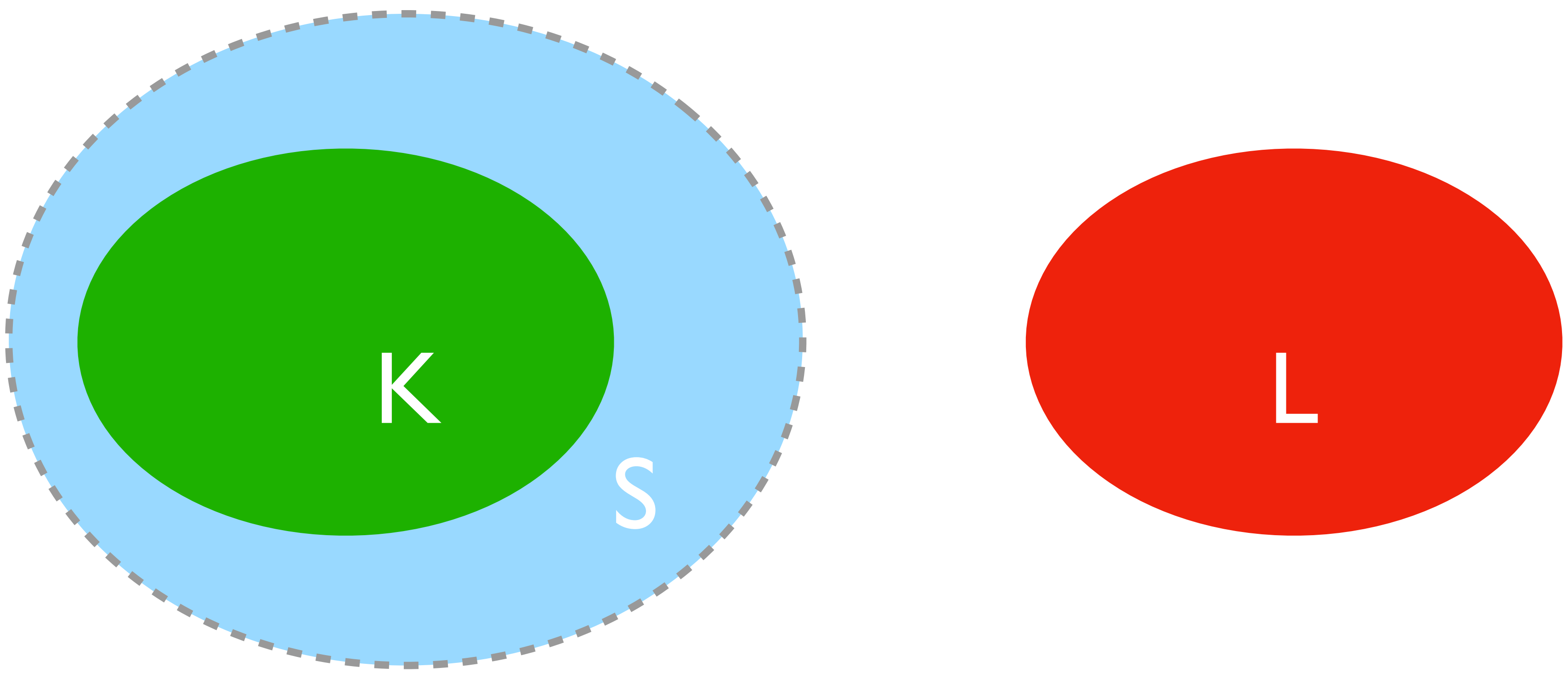


Regular-separability

Languages K and L are regular-separable
if there is a regular S such that
$K \subseteq S$ and $L \cap S = \emptyset$.

Regular-separability

Languages K and L are regular-separable
if there is a regular S such that
$K \subseteq S$ and $L \cap S = \varnothing$.



Regular-separability for VASSes

Regular-separability for VASSes

Theorem [Cz., Lasota, Meyer, Muskalla, Kumar, Saivasan]
Each two disjoint VASS languages are regular-separable.

Regular-separability for VASSes

Theorem [Cz., Lasota, Meyer, Muskalla, Kumar, Saivasan]
Each two disjoint VASS languages are regular-separable.

Theorem For each VASS V there is
a finite family of regular languages $\mathcal{F}_V$
such that if L(c_1) and L(c_2) are disjointed then
they are separable by some language from $\mathcal{F}_V$.

Proof of baby case

Proof of baby case

Fact If both L and its complement are 1-VASS languages then L is regular.

Proof of baby case

Fact If both L and its complement are 1-VASS languages then L is regular.

Higher counter values are better

Proof of baby case

Fact If both L and its complement are 1-VASS languages then L is regular.

Higher counter values are better

$\text{char}_V(w)$ = (max val in q_1, …, max val in q_n) $\in (\mathbb{N} \cup \{\perp\})^n$

Proof of baby case

Fact If both L and its complement are 1-VASS languages then L is regular.

Higher counter values are better

$\text{char}_V(w) = (\text{max val in } q_1, \ldots, \text{max val in } q_n) \in (\mathbb{N} \cup \{\bot\})^n$

$\text{char}_V(u) \preccurlyeq \text{char}_V(v)$

implies

for all $w \in \Sigma^*$ if $uw \in L(V)$ then $vw \in L(V)$

Proof cont.

Proof cont.

$\text{char}_V(u) \preccurlyeq \text{char}_V(v)$ implies

for all $w \in \Sigma^*$ if $uw \in L(V)$ then $vw \in L(V)$

Proof cont.

$char_V(u) \preccurlyeq char_V(v)$ implies

for all $w \in \Sigma^*$ if $uw \in L(V)$ then $vw \in L(V)$

Let $L = L(V_1)$, $compl(L) = L(V_2)$

Proof cont.

$\mathrm{char}_V(u) \preccurlyeq \mathrm{char}_V(v)$ implies

for all $w \in \Sigma^*$ if $uw \in L(V)$ then $vw \in L(V)$

Let $L = L(V_1)$, $\mathrm{compl}(L) = L(V_2)$

$\mathrm{char}(w) = (\mathrm{char}_{V_1}(w), \mathrm{char}_{V_2}(w))$

Proof cont.

$char_V(u) \preccurlyeq char_V(v)$ implies

for all $w \in \Sigma^*$ if $uw \in L(V)$ then $vw \in L(V)$

Let $L = L(V_1)$, $compl(L) = L(V_2)$

$char(w) = (char_{V_1}(w), char_{V_2}(w))$

$char(u) \preccurlyeq char(v)$ implies

for all $w \in \Sigma^*$ $uw \in L(V_1) \Leftrightarrow vw \in L(V_1)$

Proof cont.

$\text{char}_V(u) \preccurlyeq \text{char}_V(v)$ implies

for all $w \in \Sigma^*$ if $uw \in L(V)$ then $vw \in L(V)$

Let $L = L(V_1)$, $\text{compl}(L) = L(V_2)$

$\text{char}(w) = (\text{char}_{V_1}(w), \text{char}_{V_2}(w))$

$\text{char}(u) \preccurlyeq \text{char}(v)$ implies

for all $w \in \Sigma^*$ $uw \in L(V_1) \Leftrightarrow vw \in L(V_1)$

Comparable u and v are Myhill-Nerode related

Proof cont.

Proof cont.

char(u) $\preccurlyeq$ char(v) implies

u and v are Myhill-Nerode related

Proof cont.

char(u) $\preccurlyeq$ char(v) implies
u and v are Myhill-Nerode related

By Dickson's Lemma $\preccurlyeq$ is a well-quasi order

Proof cont.

char(u) $\preccurlyeq$ char(v) implies
u and v are Myhill-Nerode related

By Dickson's Lemma $\preccurlyeq$ is a well-quasi order

Myhill-Nerode index of $L(V_1)$ is finite

Proof cont.

char(u) $\preccurlyeq$ char(v) implies
u and v are Myhill-Nerode related

By Dickson's Lemma $\preccurlyeq$ is a well-quasi order

Myhill-Nerode index of L(V_1) is finite

Thus L(V_1) is regular

Generalizations

Generalizations

Works also for VASSes with similar order

Generalizations

Works also for VASSes with similar order

Regular-separability needs new techniques

Generalizations

Works also for VASSes with similar order

Regular-separability needs new techniques

Works also for all finitely-branching
Well Structured Transition Systems (WSTSes):

Generalizations

Works also for VASSes with similar order

Regular-separability needs new techniques

Works also for all finitely-branching
Well Structured Transition Systems (WSTSes):

wqo $\preccurlyeq$ on configurations

Generalizations

Works also for VASSes with similar order

Regular-separability needs new techniques

Works also for all finitely-branching
Well Structured Transition Systems (WSTSes):

- wqo $\preccurlyeq$ on configurations
- $\preccurlyeq$-bigger configurations can simulate smaller

Generalizations

Works also for VASSes with similar order

Regular-separability needs new techniques

Works also for all finitely-branching
Well Structured Transition Systems (WSTSes):

- wqo $\preccurlyeq$ on configurations
- $\preccurlyeq$-bigger configurations can simulate smaller
- $\preccurlyeq$-upward closed acceptance condition

Future-determinisation

Future-determinisation

Let h: $\Sigma^* \rightarrow M$

Future-determinisation

Let $h: \Sigma^* \rightarrow M$

Then $map_h: \Sigma^* \rightarrow (\Sigma \times M)^*$

Future-determinisation

Let $h: \Sigma^* \rightarrow M$ Then $map_h: \Sigma^* \rightarrow (\Sigma \times M)^*$

$$map_h(a\ b\ c) = (*, h(abc))\ (a, h(bc))\ (b, h(c))\ (c, h(\varepsilon))$$

Future-determinisation

Let $h : \Sigma^* \to M$ Then $map_h : \Sigma^* \to (\Sigma \times M)^*$

$$map_h(a\ b\ c) = (*, h(abc))\ (a, h(bc))\ (b, h(c))\ (c, h(\varepsilon))$$

Claim 1: For each $h : \Sigma^* \to M$ it holds

$$K \subseteq L \Leftrightarrow map_h(K) \subseteq map_h(L)$$

Future-determinisation

Let $h: \Sigma^* \to M$ Then $map_h: \Sigma^* \to (\Sigma \times M)^*$

$map_h(a\ b\ c) = (*, h(abc))\ (a, h(bc))\ (b, h(c))\ (c, h(\varepsilon))$

Claim 1: For each $h: \Sigma^* \to M$ it holds

$K \subseteq L \Leftrightarrow map_h(K) \subseteq map_h(L)$

Claim 2: For each $h: \Sigma^* \to M$ and each VASS V one can construct another VASS V_h accepting $map_h(L(V))$

Future-determinisation

Future-determinisation

Claim 2: For each $h: \Sigma^* \to M$ and each VASS V one can construct another VASS V_h accepting $map_h(L(V))$

Future-determinisation

Claim 2: For each $h: \Sigma^* \to M$ and each VASS V one can construct another VASS V_h accepting $map_h(L(V))$

States are pairs in
$Q \times M \cup \{(init, \$)\}$

Future-determinisation

Claim 2: For each $h: \Sigma^* \to M$ and each VASS V one can construct another VASS V_h accepting $map_h(L(V))$

States are pairs in $Q \times M \cup \{(init, \$)\}$

initial is $(init, \$)$

Future-determinisation

Claim 2: For each $h: \Sigma^* \to M$ and each VASS V one can construct another VASS V_h accepting $map_h(L(V))$

States are pairs in
$Q \times M \cup \{(init, \$)\}$

initial is $(init, \$)$

final are $F \times \{h(\varepsilon)\}$

Future-determinisation

Claim 2: For each $h: \Sigma^* \to M$ and each VASS V one can construct another VASS V_h accepting $map_h(L(V))$

States are pairs in $Q \times M \cup \{(init, \$)\}$

initial is $(init, \$)$

final are $F \times \{h(\varepsilon)\}$

For a transition

Future-determinisation

Claim 2: For each $h: \Sigma^* \to M$ and each VASS V one can construct another VASS V_h accepting $map_h(L(V))$

States are pairs in $Q \times M \cup \{(init, \$)\}$

initial is $(init, \$)$

final are $F \times \{h(\varepsilon)\}$

For a transition $p \xrightarrow[v]{a} q$

Future-determinisation

Claim 2: For each $h: \Sigma^* \rightarrow M$ and each VASS V one can construct another VASS V_h accepting $map_h(L(V))$

States are pairs in $Q \times M \cup \{(init, \$)\}$

initial is $(init, \$)$

final are $F \times \{h(\varepsilon)\}$

For a transition $p \xrightarrow[v]{a} q$

We create

Future-determinisation

Claim 2: For each $h: \Sigma^* \to M$ and each VASS V one can construct another VASS V_h accepting $map_h(L(V))$

States are pairs in
$Q \times M \cup \{(init, \$)\}$

initial is $(init, \$)$

final are $F \times \{h(\varepsilon)\}$

For a transition

$$p \xrightarrow[v]{a} q$$

We create

$$(p,m) \xrightarrow[v]{(a,m')} (q,m')$$

Future-determinisation

Claim 2: For each $h: \Sigma^* \to M$ and each VASS V one can construct another VASS V_h accepting $map_h(L(V))$

States are pairs in $Q \times M \cup \{(init, \$)\}$

initial is $(init, \$)$

final are $F \times \{h(\varepsilon)\}$

For a transition $p \xrightarrow[v]{a} q$

We create $(p,m) \xrightarrow[v]{(a,m')} (q,m')$

if $m = h(a)\, m'$ or $m = \$, a = *$

Future-determinisation

Future-determinisation

Theorem For each VASS V there is a finite family of regular languages $\mathscr{F}_V$ such that if $L(c_1)$ and $L(c_2)$ are disjointed then they are separable by some language from $\mathscr{F}_V$.

Future-determinisation

Theorem For each VASS V there is
a finite family of regular languages $\mathcal{F}_V$
such that if $L(c_1)$ and $L(c_2)$ are disjointed then
they are separable by some language from $\mathcal{F}_V$.

Claim 3: For each unambiguous VASS V and $h: \Sigma^* \rightarrow M$
recognising all the languages from $\mathcal{F}_V$
the extended VASS V'_h is deterministic and computable.

Future-determinisation

Theorem For each VASS V there is
a finite family of regular languages $\mathcal{F}_V$
such that if $L(c_1)$ and $L(c_2)$ are disjointed then
they are separable by some language from $\mathcal{F}_V$.

Claim 3: For each unambiguous VASS V and $h: \Sigma^* \to M$
recognising all the languages from $\mathcal{F}_V$
the extended VASS V'_h is deterministic and computable.

So language equivalence is decidable for uVASSes

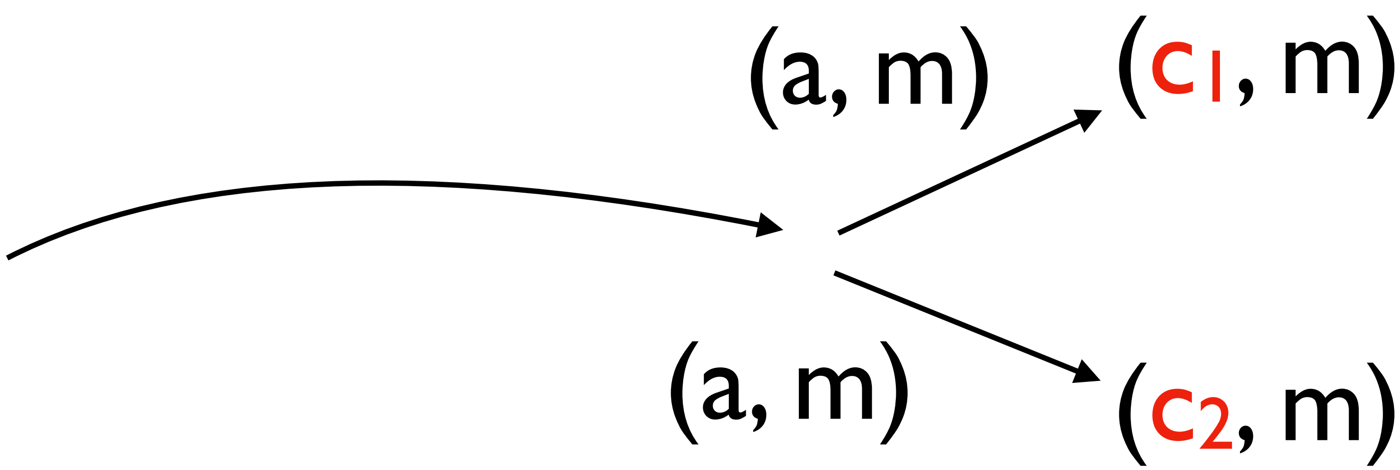
Proof

Proof

Claim 3: For each unambiguous VASS V and h: $\Sigma^* \to M$ recognising all the languages from $\mathcal{F}_V$ the extended VASS V'_h is deterministic and computable.

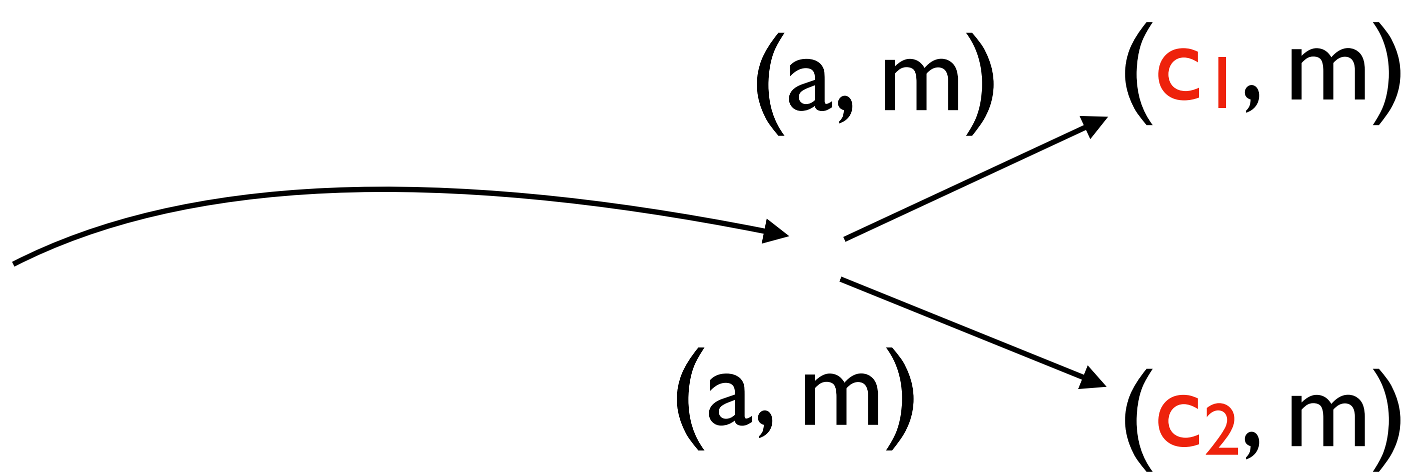
Proof

Claim 3: For each unambiguous VASS V and h: $\Sigma^* \to M$ recognising all the languages from $\mathcal{F}_V$ the extended VASS V'_h is deterministic and computable.



Proof

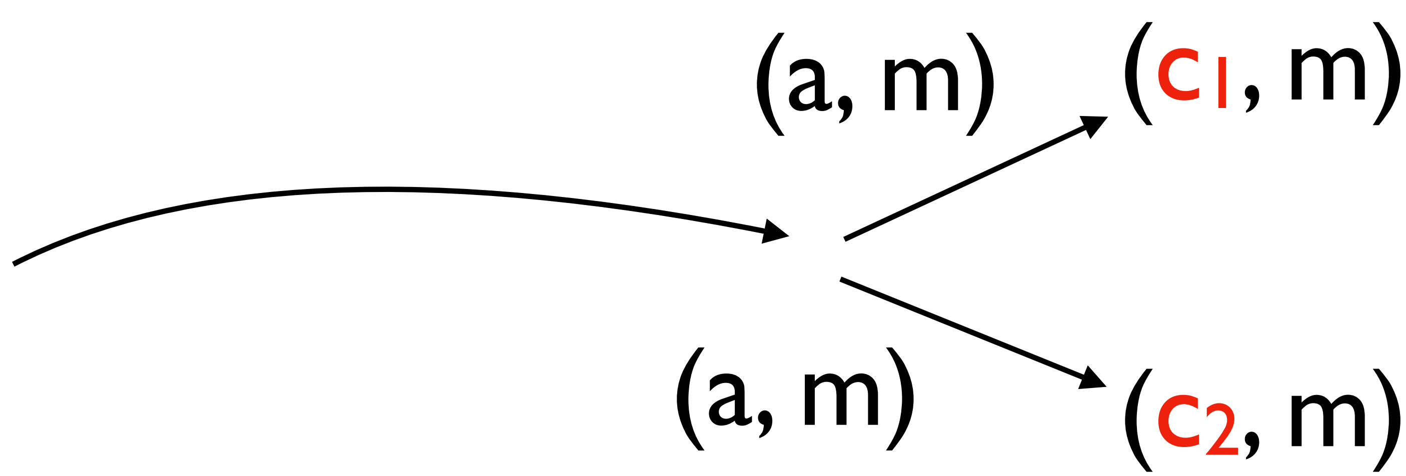
Claim 3: For each unambiguous VASS V and h: $\Sigma^* \rightarrow M$ recognising all the languages from $\mathscr{F}_V$ the extended VASS V'_h is deterministic and computable.



if $L(c_1)$ and $L(c_2)$ both nonempty then they intersect

Proof

Claim 3: For each unambiguous VASS V and $h: \Sigma^* \to M$ recognising all the languages from $\mathcal{F}_V$ the extended VASS V'_h is deterministic and computable.

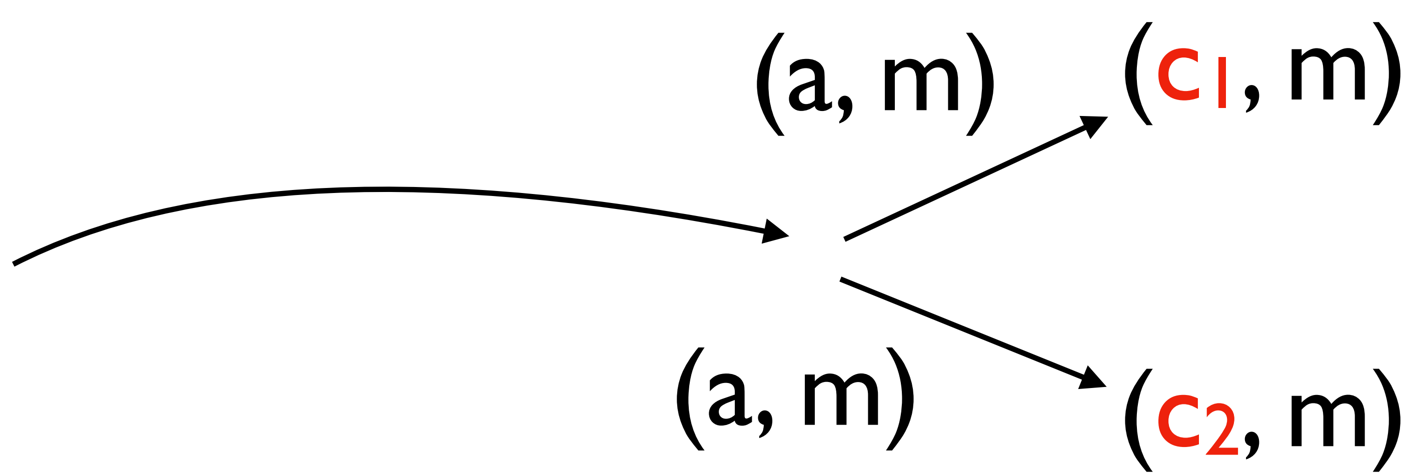


if $L(c_1)$ and $L(c_2)$ both nonempty then they intersect

this contradicts unambiguity!

Proof

Claim 3: For each unambiguous VASS V and $h: \Sigma^* \to M$ recognising all the languages from $\mathcal{F}_V$ the extended VASS V'_h is deterministic and computable.



if $L(c_1)$ and $L(c_2)$ both nonempty then they intersect

this contradicts unambiguity!

so V_h is deterministic after removing c with $L(c) = \varnothing$

Future work

Future work

Regular-separability theorem works for WSTS
What about future-determinisation?

Future work

Regular-separability theorem works for WSTS
What about future-determinisation?

Can future-determinisation help for other problems?

Future work

Regular-separability theorem works for WSTS
What about future-determinisation?

Can future-determinisation help for other problems?

Some deeper connection between
separability and unambiguity?

Future work

Future work

What about path-equivalence?

Future work

What about path-equivalence?

Can weighted models help?

Future work

What about path-equivalence?

Can weighted models help?

Thank you!