Computing a complete finite prefix of the unfolding of a 1-safe net:

McMillan / Esparza

- Every reachable marking is represented
- Every live transition occurs in the prefix

Configuration C

$\downarrow$ closed of events, #-free.

$M_{in} \overset{C}{\rightsquigarrow} M(C)$

$\Uparrow C$ = C' s.t. $C \subseteq C'$

$\forall C' \in \Uparrow C$, $C' = C + E$

$Mark(C_1) = Mark(C_2) \Rightarrow$

$\forall C_1' = C_1 + E_1 \in \Uparrow C_1$

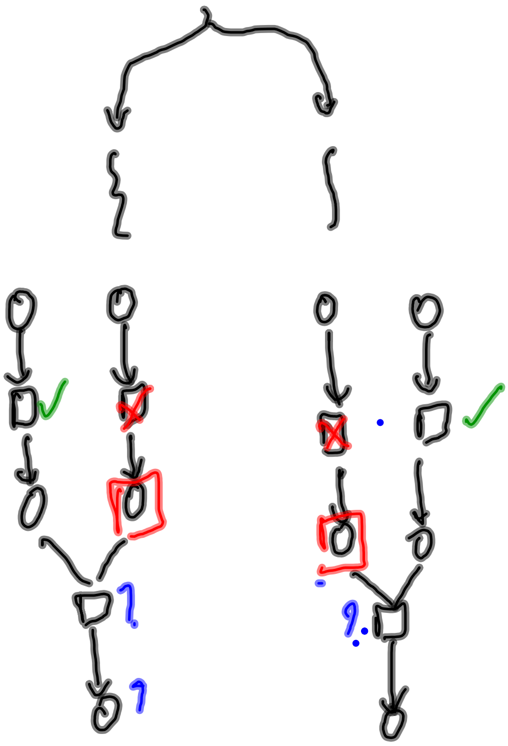
$\exists C_2' = C_2 + E_2 \in \Uparrow C_2$ s.t. $E_1 \approx E_2$

Isomorphism

Local configurations : $[e] = \downarrow e$ (e is an event in unfolding)

$Mark([e_1]) = Mark([e_2])$.

Identify one of these as a "cutoff" event - do not explore $\Uparrow e$

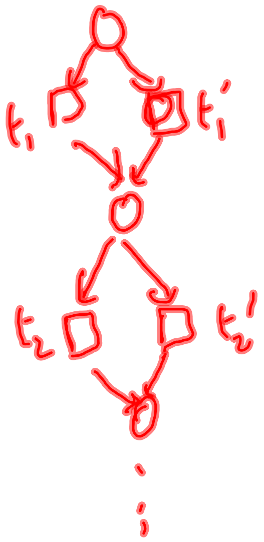


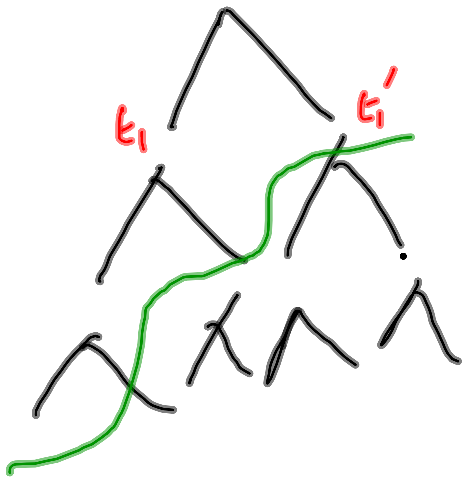
McMillan's idea

Cut off if

$Mark([e_1]) = Mark([e_2])$

& $|[e_1]| < |[e_2]|$





Adequate order: $\prec$ on configuration : p.o. st.

- $C_1 \subset C_2 \Rightarrow C_1 \prec C_2$
- $\prec$ is well founded
- $C_1 \prec C_2$ & $Mark(C_1) = Mark(C_2)$

  $\forall E_1 \quad C_1 + E_1 \prec C_2 + I(E_1)$

Unfold, at each point explore e s.t. [e] is
< minimal among enabled events

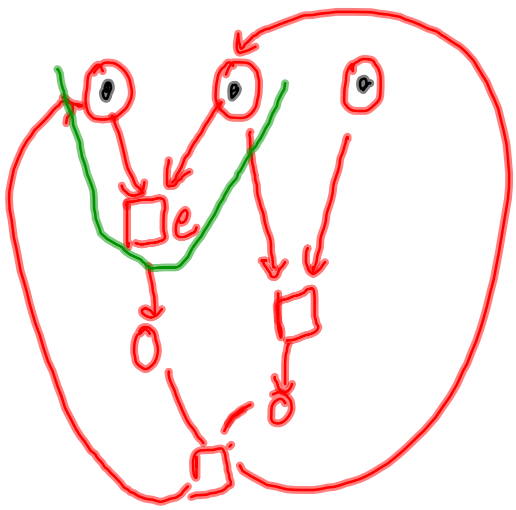
Cutoff event e' is a cutoff event if ∃e in
the prefix already constructed s.t.

$$Mark([e]) = Mark([e'])$$

$$\& \; [e] < [e']$$

This unfolding strategy generates a finite prefix
that is complete

Finiteness: Along any path, if we see 2^n+1 copies of e, two of them must agree on $Mark([e])$



m = # reachable markings

Any sequence of $m+1$ events generates a cut off event

$\Rightarrow$ depth (unfolding) $\leq m+1$

Prove that # places is finite

Completeness

Suppose $M \in Reach(M_{in})$ is not represented.

$\exists C$ in inf. unf s.t. $Mark(C) = M$

$\therefore$ C contains some cutoff event ~~E~~ e

$\therefore$ $C = [e] + E$

e cutoff by $e' \Rightarrow C' = [e'] + E$ has same marking

$C' < C$ since $[e'] < [e]$

Repeat arg for C' not present in Fin

Find $C'' < C'$ Contradiction because $<$ is well founded

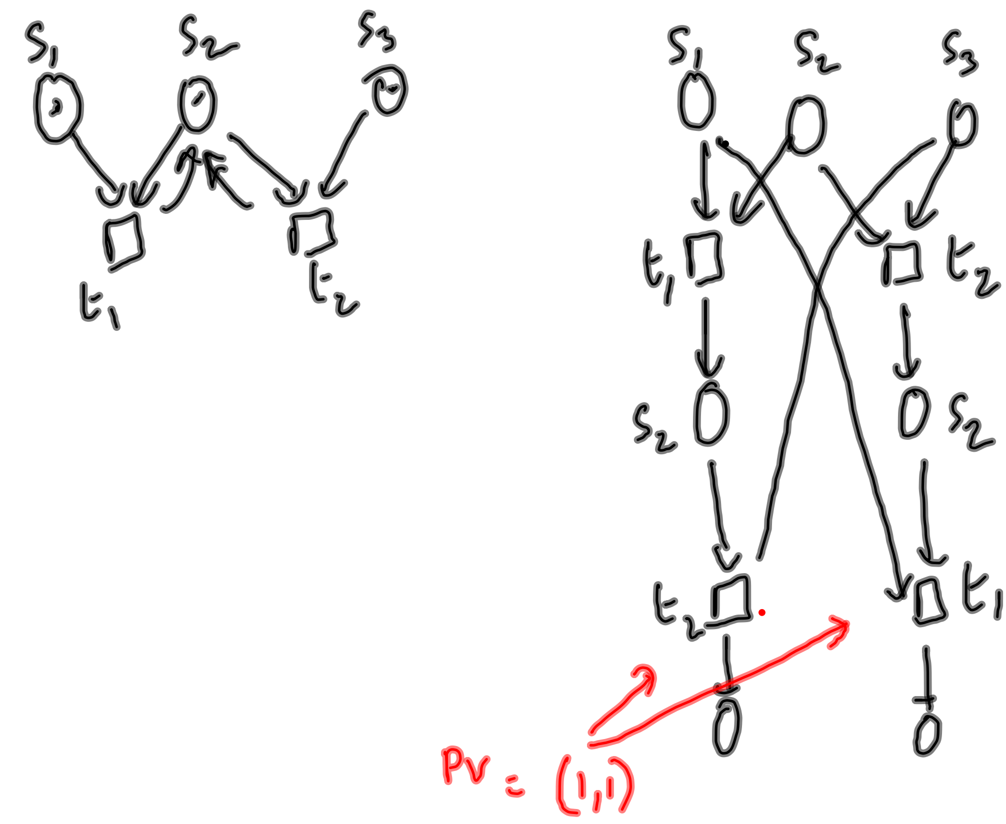
An improved adequate order

$C_1 <_E C_2$ if $|C_1| < |C_2|$

or $|C_1| = |C_2|$ &

fix a linear order on transitions in original net $\rightarrow$ ParikhVector$(C_1) <$ ParikhVector(C_2)

Claim: This is an adequate order


s1
s2
s3
t1
t2
s1
s2
s3
t1
t2
s2
s2
t2
t1
Pv = (1,1)

"Total" adequate order

Whenever we construct Fin using $\prec$.

if we explore e' & $Mark([e']) = Mark([e])$

for some existing e, then $[e] \prec [e']$

$<_E$ size   McMillan

+ Parikh Vector

$<_F$

+ Parikh Vector on max step decomposition

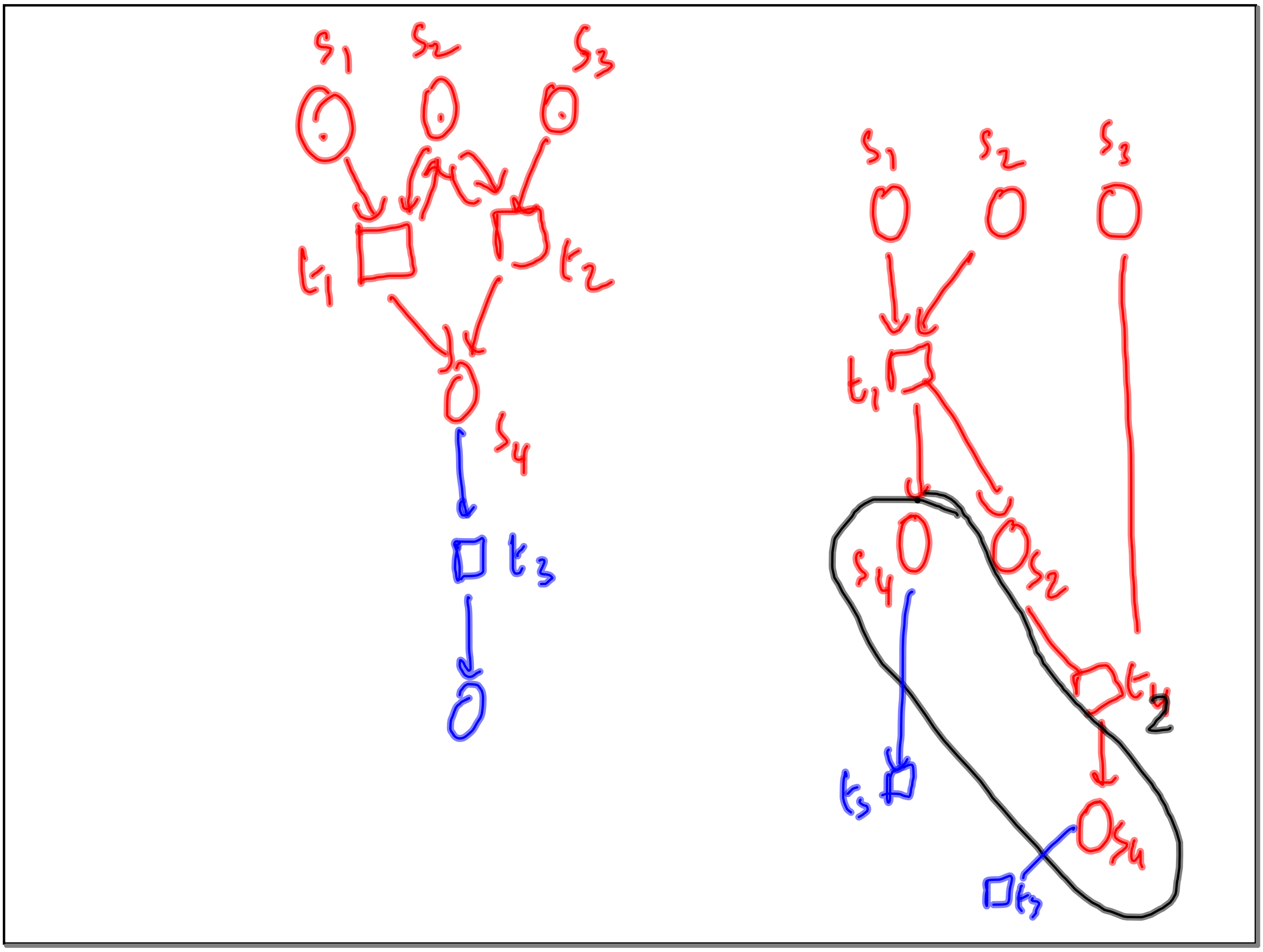
= Foata Normal Form

$T = (E, \leq, \lambda)$ Minimal element of trace = Step 1

Delete min elements & iterate

Show that $<_F$ is adequate and total

Crucially uses 1-safeness


s1
s2
s3
t1
t2
s4
t3
s1
s2
s3
t1
s4
s2
t4
2
t3
s4
t3

Timed Systems

Incorporate time into automata/languages

Abstractly: Behaviour is a word $w \in \Sigma^*$

Action a_i is done at t_i

$w = a_1 \quad a_2 \quad a_3 \quad \cdots \quad a_k$.

$t_1 \leq t_2 \leq t_3 \quad \cdots \leq t_k \in \mathbb{R}$

Infinite behaviour $\Rightarrow$ time should "progress"

$\forall r \in \mathbb{R} \ \exists_i \ t_i > r$

Timed Word: (w, σ)

Pair of sequences, or a sequence of pairs

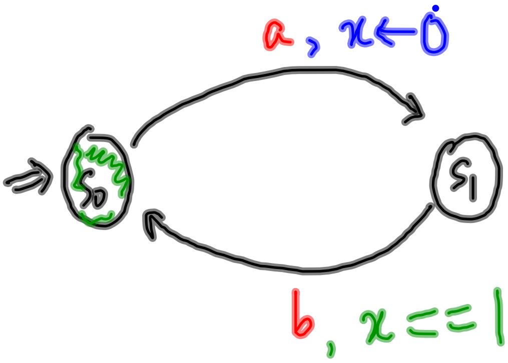
$(a_1, t_1), (a_2, t_2) \ldots (a_n, t_n)$

All timed words where each a at time t is followed by a b at time t+1

Alur & Dill Add special variables called "clocks" to finite state automata

- implicitly "grow" as time progresses
- clocks can be reset (to 0)
- conditional transitions based on clock values

$L = (ab)^*$

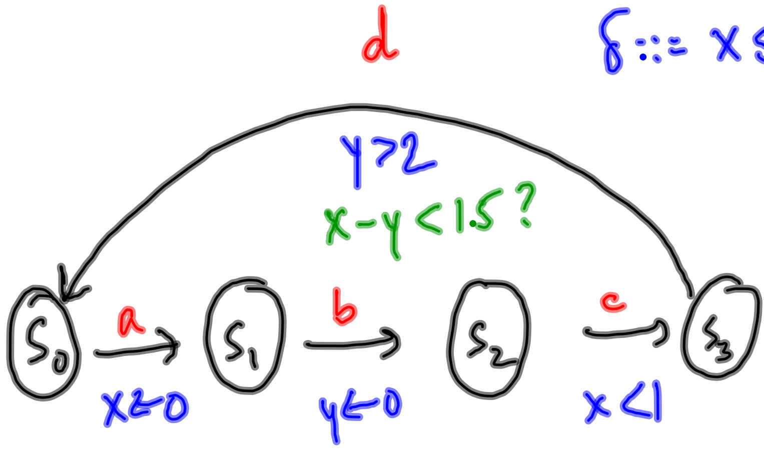


$(a, t_1), (b, t_2), (a, t_3), (b, t_4), \ldots$

s.t. $\forall$ odd i $\quad t_{i+1} - t_i = 1$

Clock constraints

$\delta ::= x \leq c \mid c \leq x \mid \neg\delta \mid \delta \wedge \delta$



Rational constants ✓

Combine clocks ? X

Syntax for conditional expressions

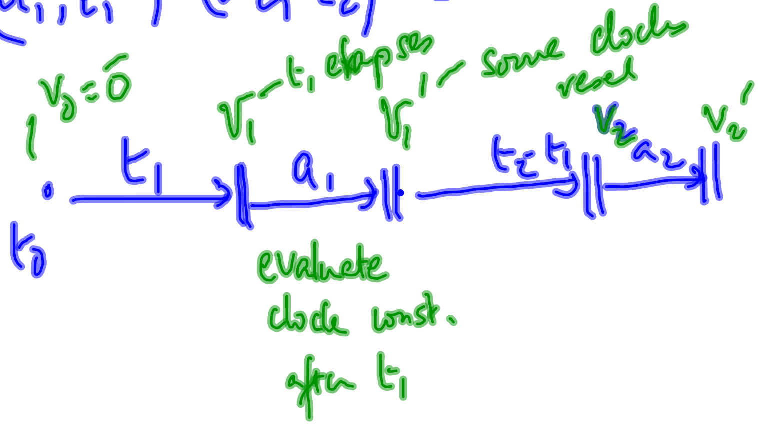
C = set of clocks (finite)

Clock valuation : $v : C \to \mathbb{R}_{\geq 0}$

$v \models \varphi$, φ is a clock constraint

Wrt clock valuation v, clock constraint φ is true

$(a_1, t_1)\ (a_2, t_2)\ \ldots$

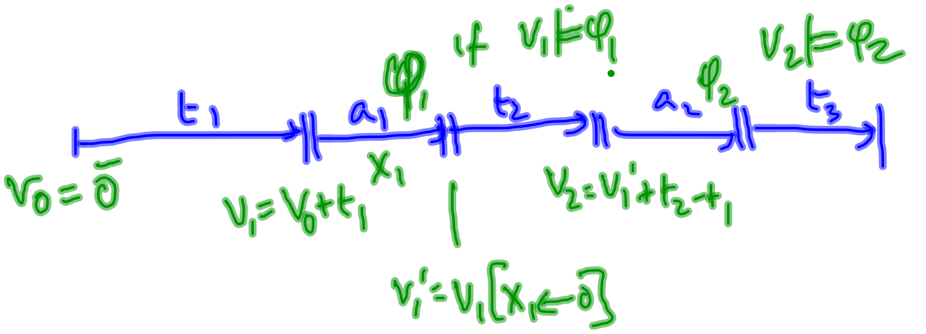


Operations on valuations

$$\underbrace{\overset{\text{old}}{\overline{v}}+t}_{\text{new valn}}(c) = v(c)+t \quad \forall c \in C$$

Given $X \subseteq C$ & $r \in \mathbb{R}_{\geq 0}$

$$v[X \leftarrow r](c) = \begin{cases} v(c) & \text{if } c \notin X \\ r & \text{if } c \in X \end{cases}$$



<u>Timed Automaton</u> over Σ

$TA = (Q, Q_{in}, F, C, \rightarrow)$

set of constraints wrt a set of clocks C

$\underset{\text{finite}}{\rightarrow} \subseteq \underset{\text{from}}{Q} \times \underset{\text{to}}{Q} \times \underset{\text{act}}{\Sigma} \times \underset{\text{reset}}{2^C} \times \underset{\text{constraint}}{\Phi(C)}$

$q \xrightarrow[\underset{\text{reset}}{X}]{\overset{\text{act}}{a}, \overset{\text{const.}}{\varphi}} q'$

Run of TA on a timed word (w, σ) $= (a_1, t_1)\ (a_2, t_2) \ldots (a_n, t_n)$

↓

Sequence of configurations

$$(q_{in}, \bar{0}) \xrightarrow[t_1]{a_1} (q_1, v_1) \xrightarrow[t_2]{a_2} (q_2, v_2)$$

$\in Q_{in}$ zero valuation

if $\exists$ $q_{in} \xrightarrow[X]{a_1, \varphi} q_1$

s.t. $v_1' = \bar{0} + t_1 \models \varphi$

$v_1 = v_1'[X \leftarrow 0]$

$q_1 \xrightarrow[X']{a_2, \varphi'} q_2$

$v_2' = v_1 + (t_2 - t_1) \models \varphi'$

$v_2 = v_2'[X' \leftarrow 0]$

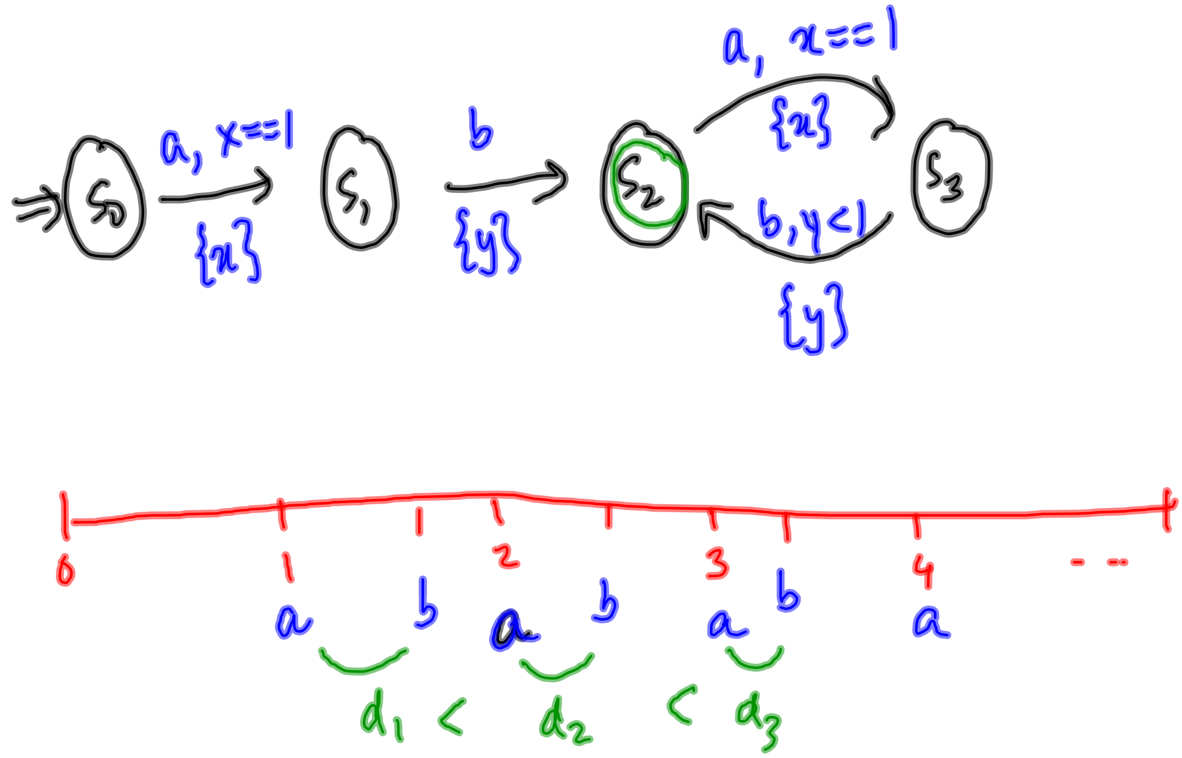
$$(q_0, \bar{0}) \xrightarrow[t_1]{a_1} (q_1, v_1) \rightarrow \cdots \xrightarrow[t_k]{a_k} (q_k, v_k)$$

Run is accepting if $q_k \in F$

$L(TA) = \{ (w, \sigma) \mid TA$ has an acc. run on $(w, \sigma) \}$

A set of timed words L is a timed regular lang if $\exists$ TA s.t. $L = L(TA)$

Time is global



Closure under

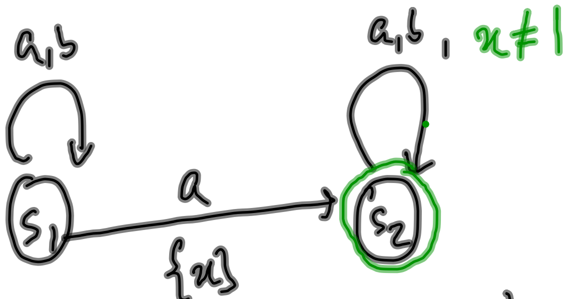
Union ✓

Intersection Product over $C_1 \cup C_2$

$$q_1 \xrightarrow[X_1]{a,\ \varphi_1} q_1' \qquad q_2 \xrightarrow[X_2]{a,\ \varphi_2} q_2'$$

$$(q_1, q_2) \xrightarrow[X_1 \cup X_2]{a\ ,\ \varphi_1 \wedge \varphi_2} (q_1', q_2')$$

Not closed under complementation

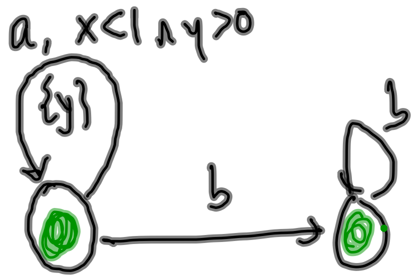


$L = \exists a$ s.t no action happens 1 unit after a

$\overline{L} = \forall a\ \exists$ action 1 unit after a

L' of the form $a^* b^*$

- all a's happen before $t=1$
- no two a events happen at the same time



Untime(L) = strip timing from set of timed words

$(w,\sigma) \longrightarrow w$

Claim: For all timed regular lang L, Untime(L) is regular

Use this to show $\bar{L}$ is not timed regular