Weighted Automata and Concurrency

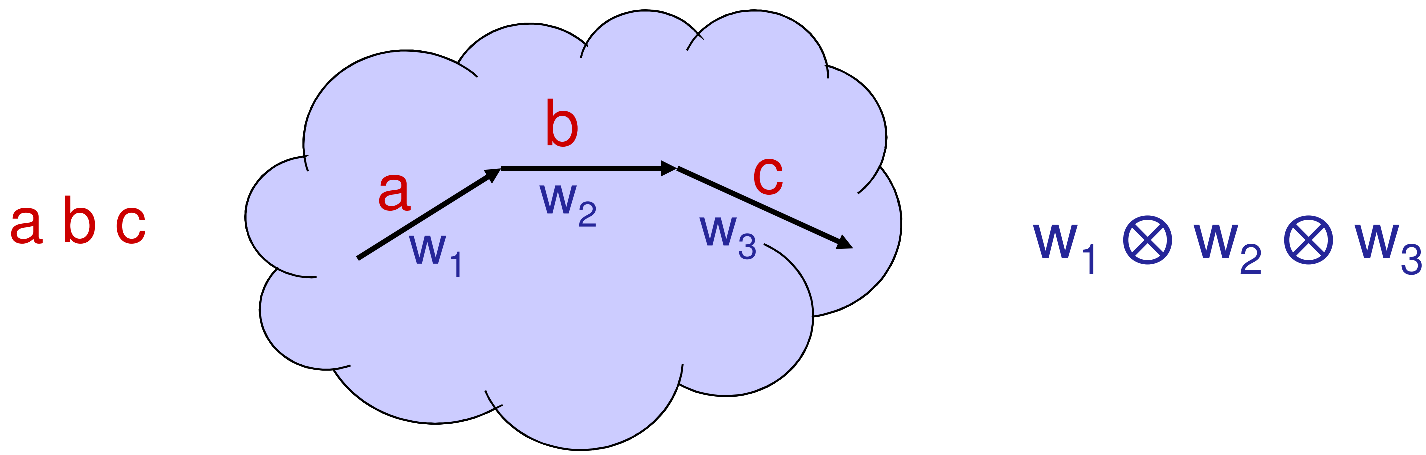
Akash Lal
Microsoft Research, India

Tayssir Touili, Nicholas Kidd and Tom Reps

ACTS II, Chennai Mathematical Institute

Weighted Automata

- A finite-state machine with weights



- A normal FSM: word $\rightarrow$ Bool
- Weighted Automata: word $\rightarrow$ Weight

Outline

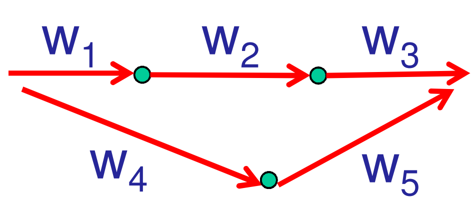
- Define weights and weighted automata
- Intersecting weighted automata
- Application
  - Generalizes to composition of weighted transducers
  - Context-Bounded Analysis: Interprocedural dataflow analysis of concurrent programs, under a bound on the number of context switches

Earlier talks

What are Weights?

- Weights == Dataflow transformers
  - Technically, they are elements of a semiring

Semiring	Dataflow Analysis
D : set of weights	DataFacts $\rightarrow$ DataFacts
$\otimes$: extend D x D $\rightarrow$ D	*Compose* (extends paths) $\tau_1 \otimes \tau_2 = \tau_2 \circ \tau_1$
$\oplus$: combine D x D $\rightarrow$ D	*Meet* (combines paths) $\tau_1 \oplus \tau_2 = \lambda d.\ \tau_1(d) \sqcap \tau_2(d)$



$$(w_1 \otimes w_2 \otimes w_3) \oplus (w_4 \otimes w_5)$$

Weighted Automata

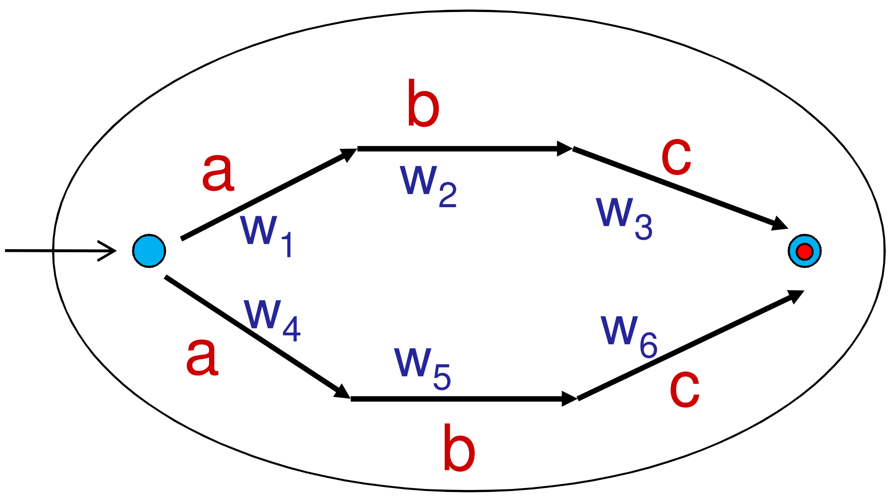
Definition 2. *A bounded idempotent semiring (or "weight domain") is a tuple* $(D, \oplus, \otimes, \overline{0}, \overline{1})$*, where* D *is a set of* weights, $\overline{0}, \overline{1} \in D$*, and* $\oplus$ *(combine) and* $\otimes$ *(extend) are binary operators on* D *such that*

1. $(D, \oplus)$ *is a commutative monoid with* $\overline{0}$ *as its neutral element, and where* $\oplus$ *is idempotent.* $(D, \otimes)$ *is a monoid with the neutral element* $\overline{1}$.
2. $\otimes$ *distributes over* $\oplus$*, i.e., for all* $a, b, c \in D$ *we have*
$$a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c) \text{ and } (a \oplus b) \otimes c = (a \otimes c) \oplus (b \otimes c) \ .$$
3. $\overline{0}$ *is an annihilator with respect to* $\otimes$*, i.e., for all* $a \in D$, $a \otimes \overline{0} = \overline{0} = \overline{0} \otimes a$.
4. *In the partial order* $\sqsubseteq$ *defined by* $\forall a, b \in D,\ a \sqsubseteq b$ *iff* $a \oplus b = a$*, there are no infinite descending chains.*

Note: extend need not be commutative

Weighted Automata

- A: word $\longrightarrow$ D
- A(s) = combine of weights of all accepting paths for s
- A(s) = $\bigoplus$ { $v(\sigma)$ | σ is an accepting path for s }



$$A(abc) = (w_1 \otimes w_2 \otimes w_3) \oplus (w_4 \otimes w_5 \otimes w_6)$$

Weighted Automata

- A(s) = $\bigoplus$ { $v(\sigma)$ | σ is an accepting path for s }

A	A(s)
(Bool, $\otimes$ is conj, $\oplus$ is disj) "true" on all edges	True iff s is accepted
(Nat, $\otimes$ is plus, $\oplus$ is min) "1" on all edges	Length of shortest accepting path
(Distributive) Dataflow Analysis	Meet-Over-All-(accepting)-Paths

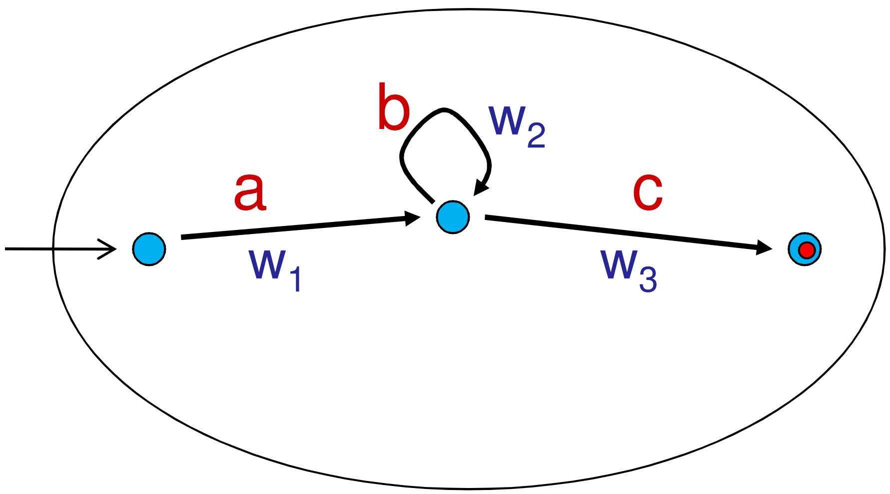
- A(T) = $\bigoplus$ { $v(\sigma)$ | σ is an accepting path for s ∈ T }

  $\bigoplus$ { A(s) | s ∈ T }

Weighted Automata

- Computing A(T)

$$A(ab^*c) = \bigoplus_i \{ w_1 \otimes w_2^i \otimes w_3 \}$$
$$= w_1 \otimes (\bigoplus_i w_2^i) \otimes w_3$$



$A(ab^*c) = (w_1 . w_2^* . w_3)$

$x . y = x \otimes y$

$x^* = (\bigoplus_i x^i)$

$(x \mid y) = x \oplus y$

Weight domain properties:

- Distributivity: $x \otimes (y \oplus z) = (x \otimes y) \oplus (x \otimes z)$
- Boundedness: All iterations x^* converge

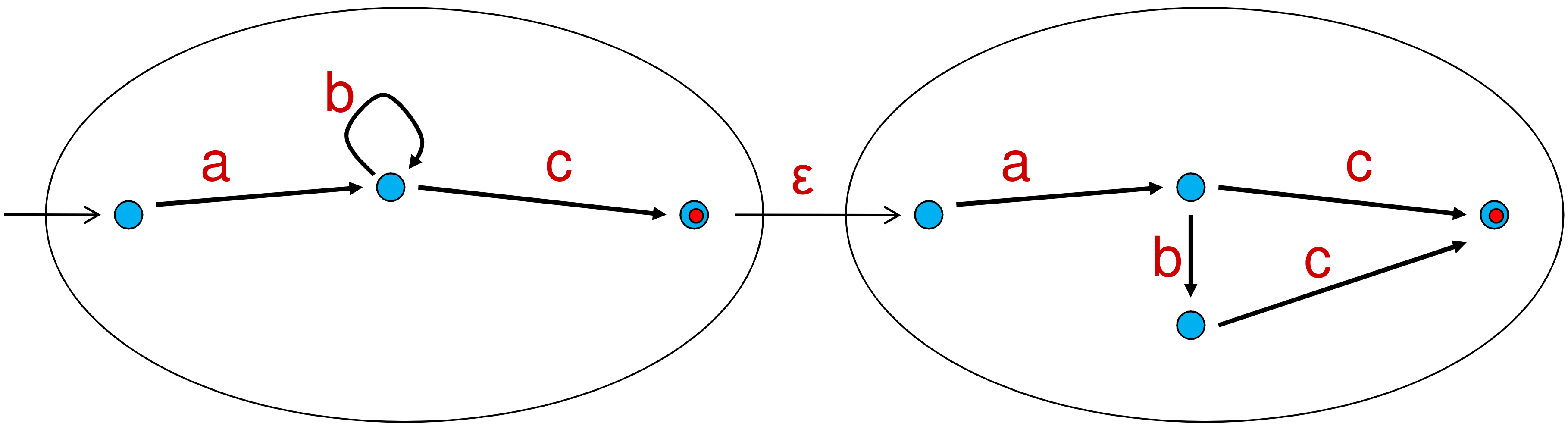
Weighted Automata Intersection

- Given A_1 and A_2, construct A_3 such that for all s,

  $A_3(s) = A_1(s) \otimes A_2(s)$
- If weight domain is (Bool, $\otimes$ is conj, $\oplus$ is disj) then
  - $A_3 = (A_1 \cap A_2)$

Weighted Automata Intersection

- $A_3(s) = A_1(s) \otimes A_2(s)$

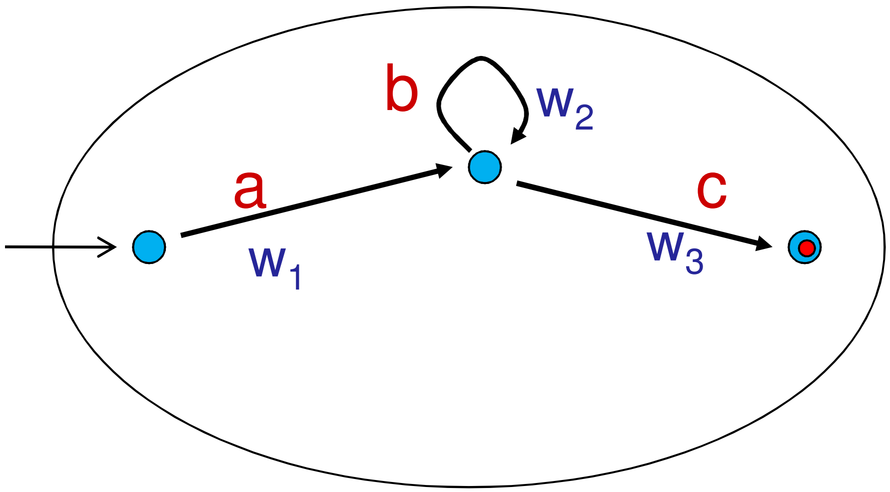


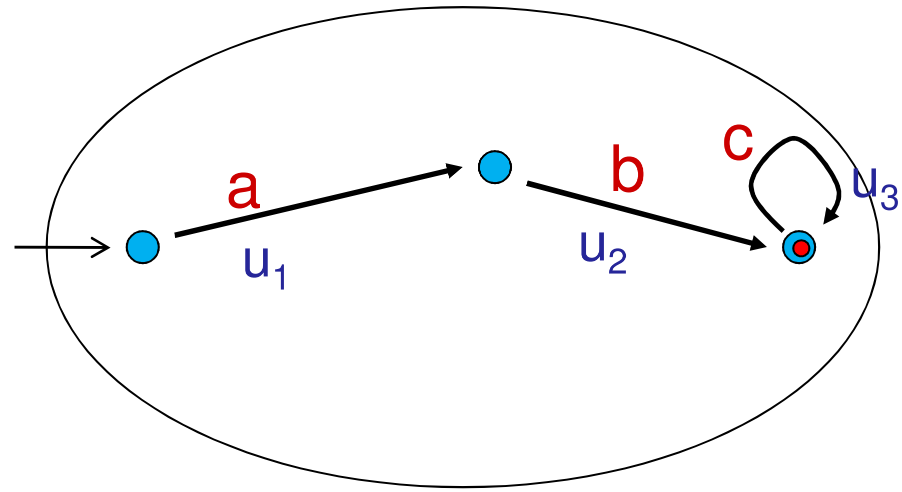
$$A_3(T) = \bigoplus \{ A_3(s) \mid s \in T \}$$
$$= \bigoplus \{ A_1(s) \otimes A_2(s) \mid s \in T \}$$
$$\neq A_1(T) \otimes A_2(T)$$

Given a regular set T, { (s s) | s ∈ T } is not regular

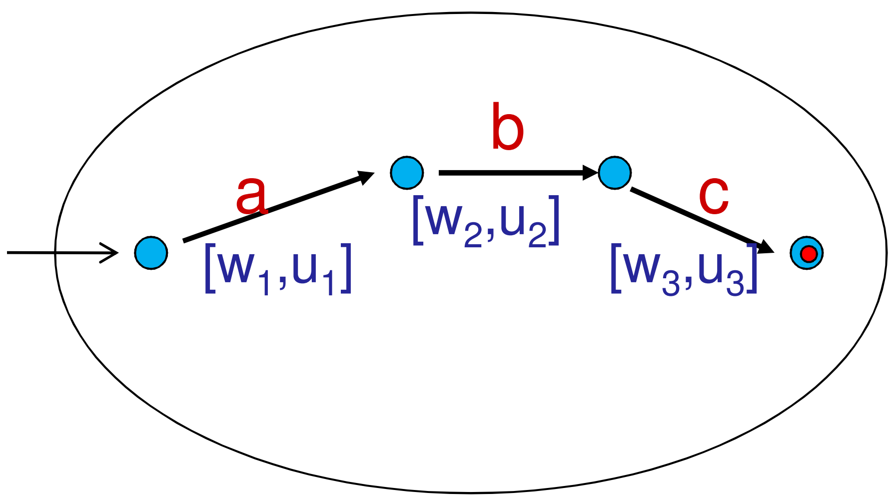
Weighted Automata Intersection

- $\forall s, A_3(s) = A_1(s) \otimes A_2(s)$



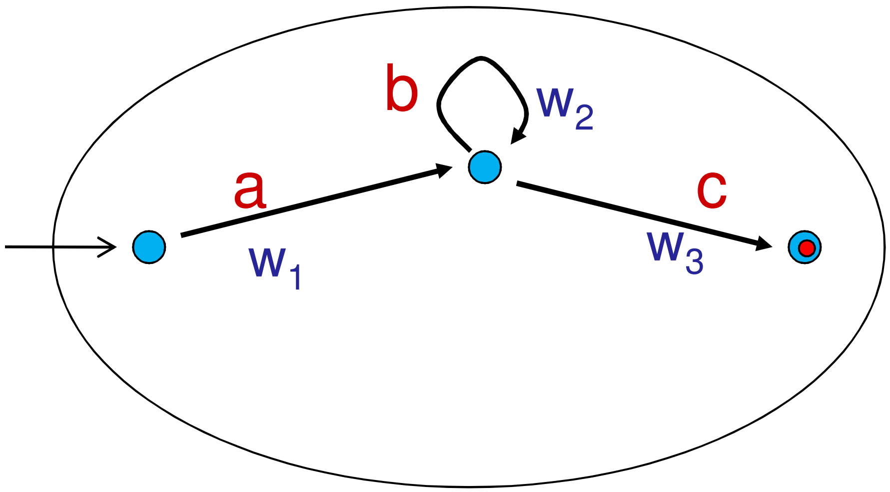


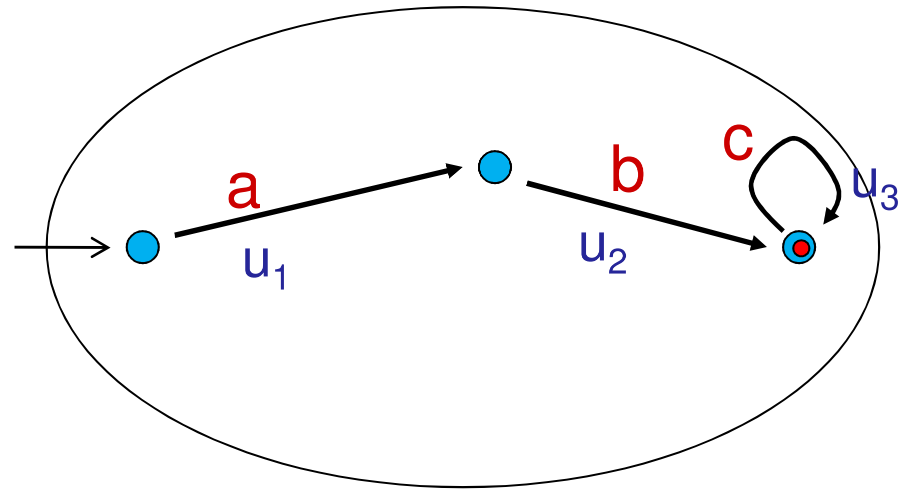
$A_3(abc) = (w_1 \otimes w_2 \otimes w_3 \otimes u_1 \otimes u_2 \otimes u_3)$



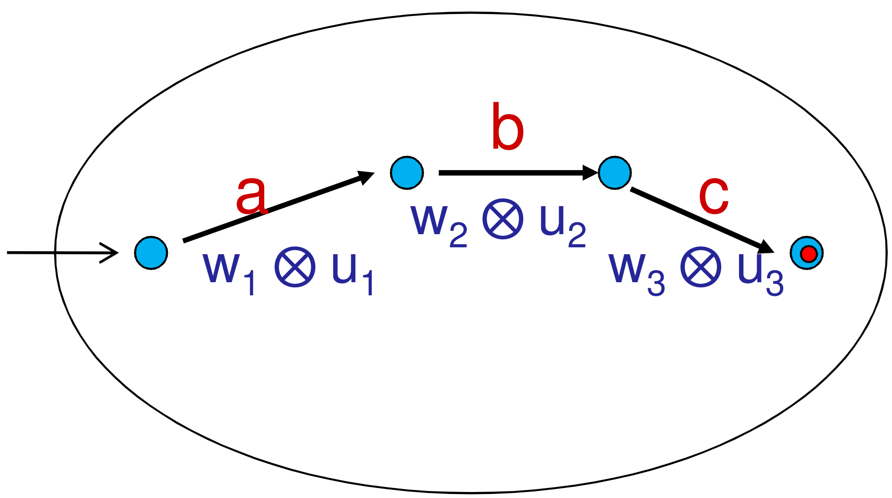
Weighted Automata Intersection

- $\forall s, A_3(s) = A_1(s) \otimes A_2(s)$





$A_3(\text{abc}) = (w_1 \otimes w_2 \otimes w_3 \otimes u_1 \otimes u_2 \otimes u_3)$



$(w_1 \otimes u_1 \otimes w_2 \otimes u_2 \otimes w_3 \otimes u_3)$

Tensor Product

- Given semiring $(D, \otimes, \oplus)$, construct a new semiring $(D_T, \otimes, \oplus)$ to represent pairs of weights from D

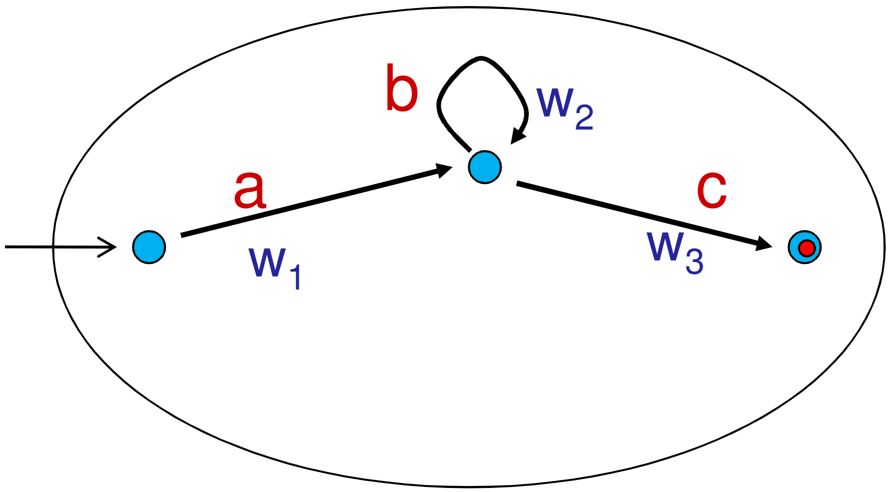
Tensor: $D \times D \rightarrow D_T$ DeTensor: $D_T \rightarrow D$

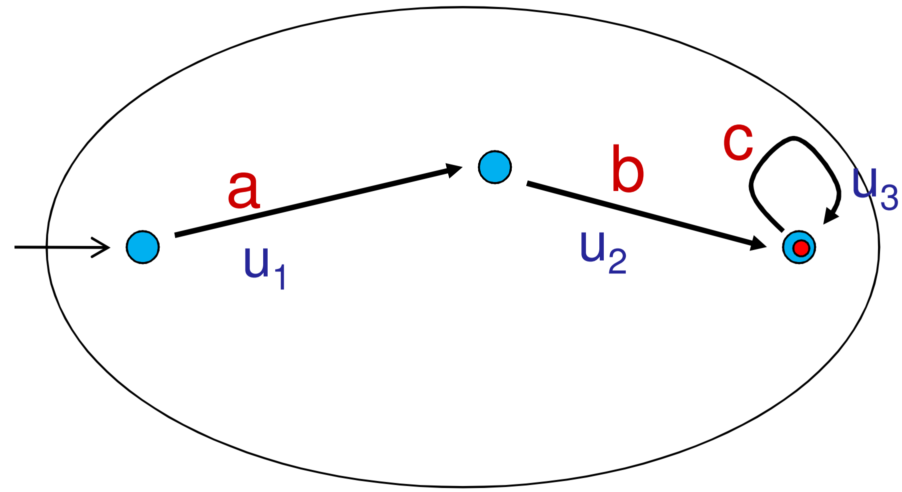
1. $\text{Tensor}(w_1,w_2) \otimes \text{Tensor}(w_3,w_4) = \text{Tensor}(w_1 \otimes w_3, w_2 \otimes w_4)$
2. $\text{DeTensor}(\text{Tensor}(w_1,w_2)) = w_1 \otimes w_2$
3. $\text{DeTensor}(W_1 \oplus W_2) = \text{DeTensor}(W_1) \oplus \text{DeTensor}(W_2)$

Note that D_T can be much bigger than $D \times D$

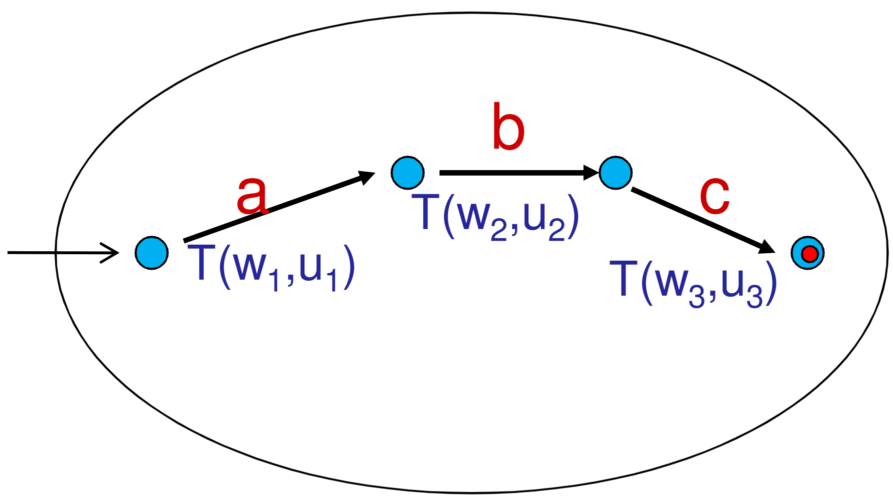
Weighted Automata Intersection

- $\forall s,\ A_3(s) = A_1(s) \otimes A_2(s)$



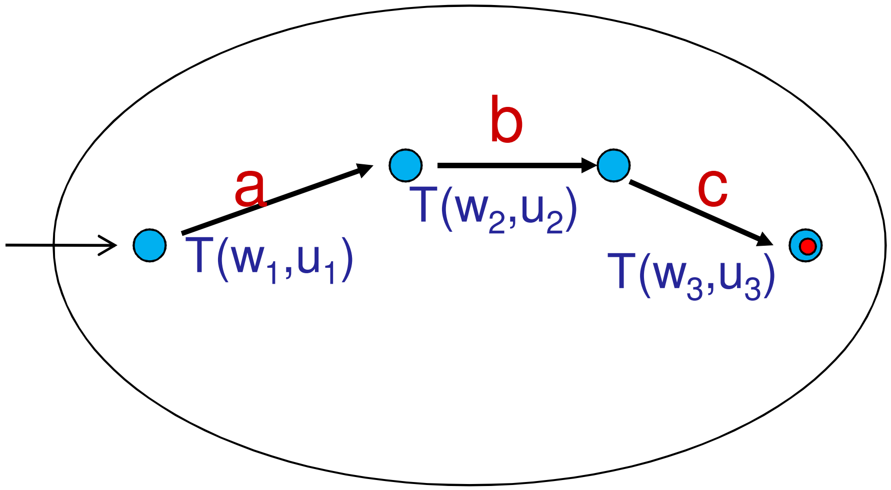


$A_3(abc) = (w_1 \otimes w_2 \otimes w_3 \otimes u_1 \otimes u_2 \otimes u_3)$



Weighted Automata Intersection

$A_3(abc) = (w_1 \otimes w_2 \otimes w_3 \otimes u_1 \otimes u_2 \otimes u_3)$



$T(w_1,u_1) \otimes T(w_2,u_2) \otimes T(w_3,u_3)$
$= T(w_1 \otimes w_2 \otimes w_3,\ u_1 \otimes u_2 \otimes u_3)$

DeTensor

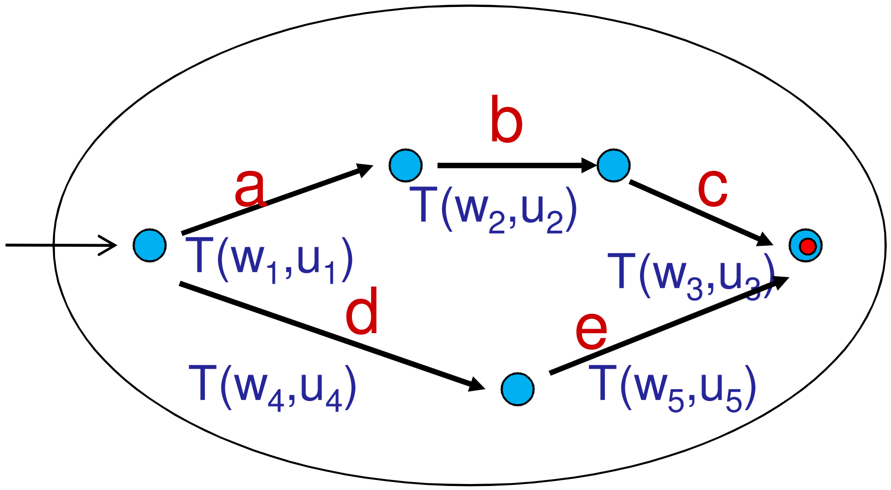
$(w_1 \otimes w_2 \otimes w_3 \otimes u_1 \otimes u_2 \otimes u_3)$

$\text{Tensor}(w_1,w_2) \otimes \text{Tensor}(w_3,w_4) = \text{Tensor}(w_1 \otimes w_3,\ w_2 \otimes w_4)$

$\text{DeTensor}(\text{Tensor}(w_1,w_2)) = w_1 \otimes w_2$

Weighted Automata Intersection

$A_3(\{abc, de\}) = (w_1 \otimes w_2 \otimes w_3 \otimes u_1 \otimes u_2 \otimes u_3) \oplus (w_4 \otimes w_5 \otimes u_4 \otimes u_5)$



$T(w_1,u_1) \otimes T(w_2,u_2) \otimes T(w_3,u_3) \oplus T(w_4,u_4) \otimes T(w_5,u_5)$

$= T(w_1 \otimes w_2 \otimes w_3, u_1 \otimes u_2 \otimes u_3) \oplus T(w_4 \otimes w_5, u_4 \otimes u_5)$

DeTensor

$(w_1 \otimes w_2 \otimes w_3 \otimes u_1 \otimes u_2 \otimes u_3) \oplus (w_4 \otimes w_5 \otimes u_4 \otimes u_5)$

$\mathrm{Tensor}(w_1,w_2) \otimes \mathrm{Tensor}(w_3,w_4) = \mathrm{Tensor}(w_1 \otimes w_3, w_2 \otimes w_4)$

$\mathrm{DeTensor}(\mathrm{Tensor}(w_1,w_2)) = w_1 \otimes w_2$

$\mathrm{DeTensor}(W_1 \oplus W_2) = \mathrm{DeTensor}(W_1) \oplus \mathrm{DeTensor}(W_2)$

Weighted Automata Intersection

<u>Theorem</u>: For any set of words T,

$$\text{DeTensor}(A_3(T)) = \bigoplus \{A_1(s) \otimes A_2(s) \mid s \in T\}$$

$$\text{DeTensor}\left(\bigoplus \{ A_3(s) \mid s \in T \} \right) =$$

$$\bigoplus \{ \text{DeTensor}(A_3(s)) \mid s \in T \} =$$

$$\bigoplus \{ \text{DeTensor}(\text{Tensor}(A_1(s),A_2(s))) \mid s \in T \} =$$

$$\bigoplus \{A_1(s) \otimes A_2(s) \mid s \in T \}$$

Tensors

- Tensors are good, but do they exist?
  - Yes!
- If (D, $\otimes$) is commutative:
  - Then D_T = D, Tensor(w_1,w_2) = $w_1 \otimes w_2$, DeTensor is identity
- If D is the set of matrices over a commutative domain
  - Extend is matrix multiplication, combine is point-wise
  - Tensor is Kronecker product

Tensors

- Kronecker product

a_1	a_2
a_3	a_4

$\odot$

b_1	b_2
b_3	b_4

$=$

a_1b_1	a_1b_2	a_2b_1	a_2b_2
a_1b_3	a_1b_4	a_2b_3	a_2b_4
a_3b_1	a_3b_2	a_4b_1	a_4b_2
a_3b_3	a_3b_4	a_4b_3	a_4b_4

↓ DeTensor

$a_1b_1 + a_2b_3$	$a_1b_2 + a_2b_4$
$a_3b_1 + a_4b_3$	$a_3b_2 + a_4b_4$

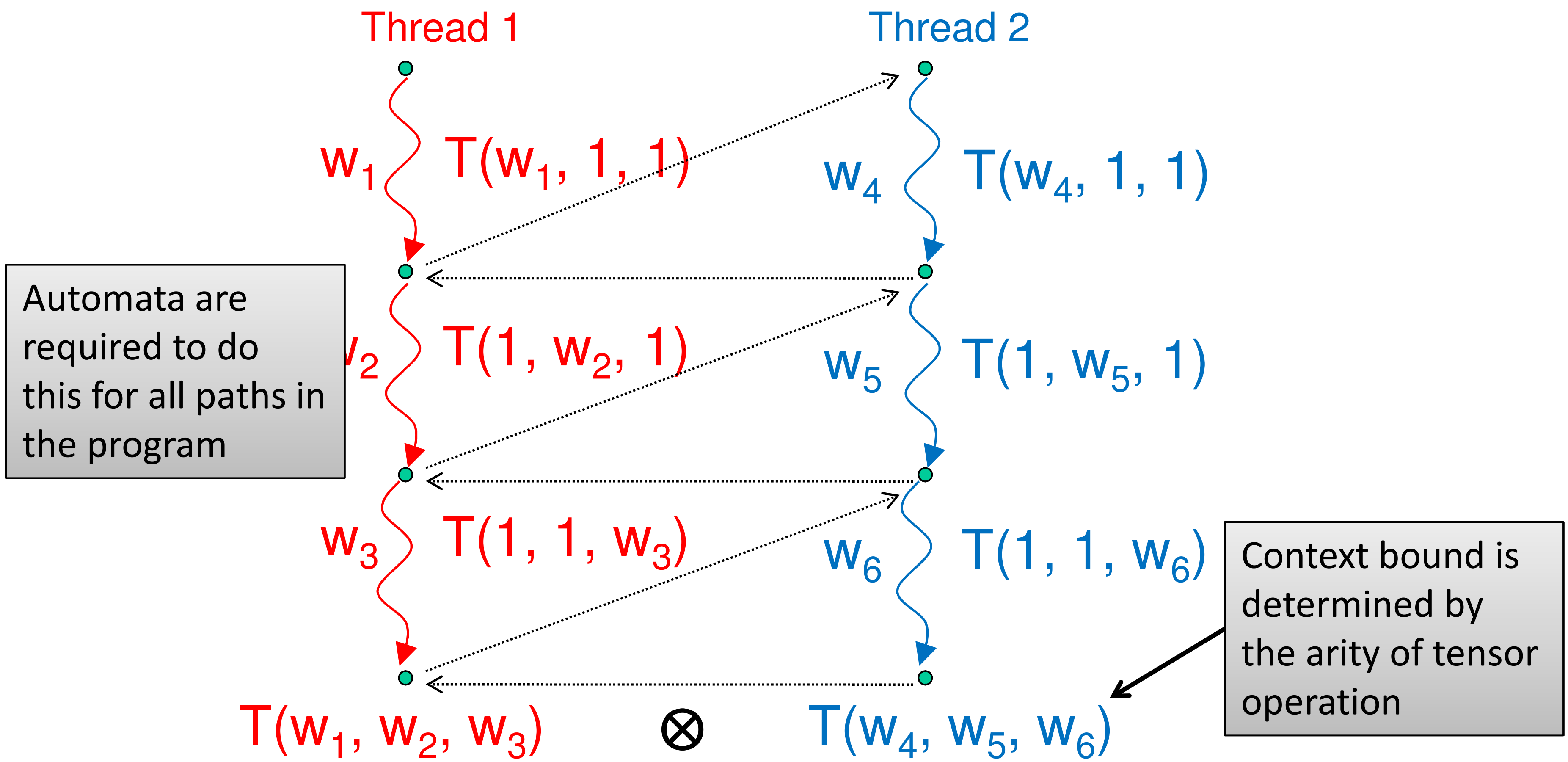
Tensors

- D is the set of matrices over a commutative domain
  - Finite relations (matrices over Booleans)
  - Affine relations (matrices over integers)
- Q: Does tensor product exist for all (bounded idempotent) semirings?

Part II: Context-Bounded Analysis

Tensors and Concurrency

Tensors give the necessary shuffling for interleaved executions



$$T(w_1 \otimes w_4, w_2 \otimes w_5, w_3 \otimes w_6) \rightarrow w_1 \otimes w_4 \otimes w_2 \otimes w_5 \otimes w_3 \otimes w_6$$

Application: Context-Bounded Analysis

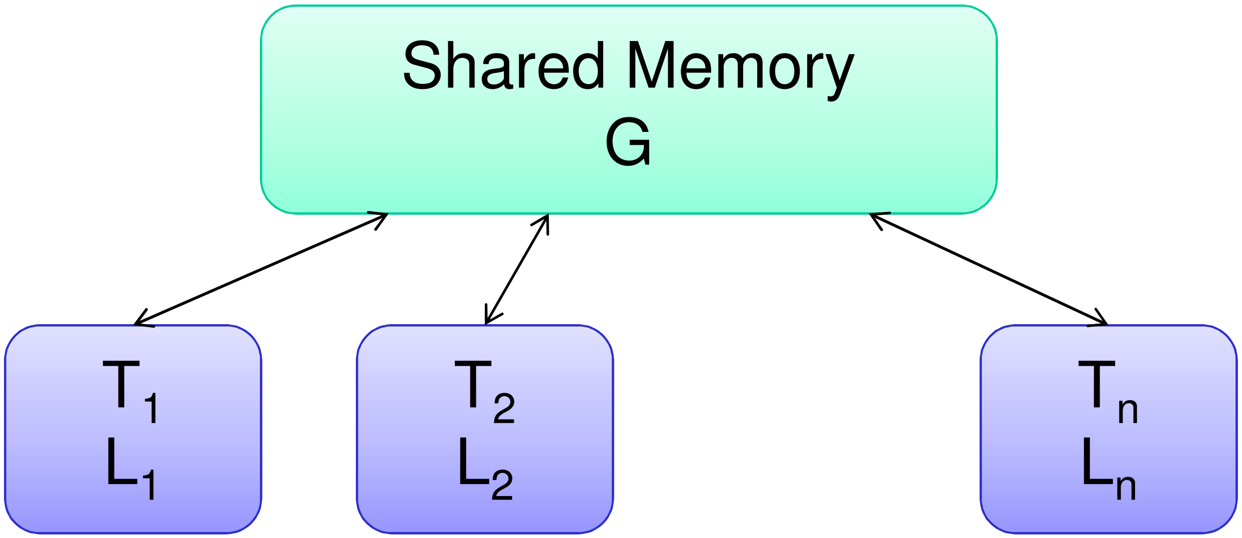
- Context Bounded Analysis: interprocedural analysis of concurrent programs under a bound the number of context switches

- Weighted Pushdown System: A PDS with weights on rules.
  - Natural model for recursive programs

- Theorem: If all threads are modeled using WPDSs, and the weight domain has a tensor product, then for any bound K, one can precisely compute MOP.
  - Can solve reachability previsely
  - Can solve dataflow analysis precisely

Context-bounded analysis

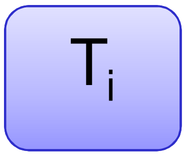
- Abstract model



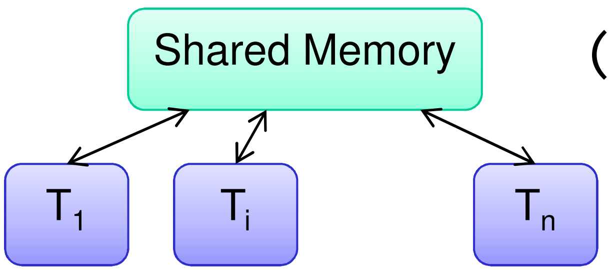
$$G \times L_1 \times L_2 \times \ldots \times L_n$$

Context-bounded analysis

- Transition Systems



$$(g, l_i) \rightarrow_{Ti} (g', l_i')$$



$$(g, l_1, \ldots, l_i, \ldots, l_n) \Rightarrow_{Ti} (g', l_1, \ldots, l_i', \ldots, l_n)$$

- Transition system for an execution context

$\Rightarrow^c$ equals $\Rightarrow_{T1}^* \cup \Rightarrow_{T2}^* \cup \ldots \cup \Rightarrow_{Tn}^*$

Context-bounded analysis

- Want to check reachability in the transition system:

$$\underbrace{\Rightarrow^{c} \Rightarrow^{c} \Rightarrow^{c} \;\ldots\; \Rightarrow^{c} \Rightarrow^{c}}_{\text{k+1 times}}$$

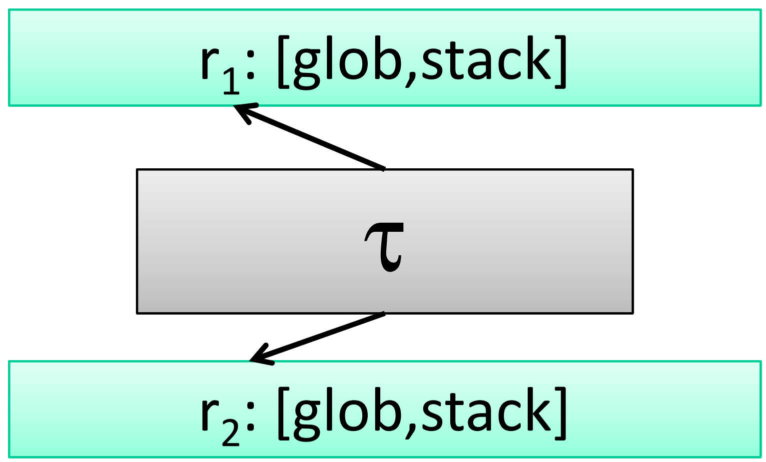
Thread Summarization

- In interprocedural analysis
  - Procedure re-analyzed for each input
  - Instead, one can build a summary
- We create a summary of an entire thread
  - Mapping starting states (input) to reachable states (output)
- Transducers: FSMs with an input and a output tape

Thread Summarization

- Reachability in a PDS can be modeled using a transducer [Caucal '92]

$(g_1,l_1) \rightarrow_T^* (g_2,l_2)$ iff $((g_1,l_1), (g_2,l_2)) \in L(\tau)$



- Advantage: transducers can be composed

$(r_1,r_2) \in L(\tau_1)$ and $(r_2,r_3) \in L(\tau_2)$ then $(r_1,r_3) \in L(\tau_1 ; \tau_2)$

Thread Summarization

For:	Construct:
$(g, l_i) \rightarrow_{Ti}^* (g', l_i')$	τ_i
$(g, l_1, \ldots, l_i, \ldots, l_n) \Rightarrow_{Ti}^* (g', l_1, \ldots, l_i', \ldots, l_n)$	τ_i^e
$\Rightarrow^c$ equals $\Rightarrow_{T1}^* \cup \ldots \cup \Rightarrow_{Tn}^*$	$\tau_c = \tau_1^e \cup \tau_2^e \cup \ldots \cup \tau_n^e$
$\Rightarrow^c \ldots \Rightarrow^c \Rightarrow^c$	$\tau_c ; \ldots ; \tau_c ; \tau_c$

Thread Summarization

- Context-bounded analysis reduces into a membership query on a transducer

- We'll extend these results to Weighted PDSs
  - Constructing weighted transducers
  - Composing weighted transducers

- Weighted Transducer: Given an input word s_1, the transducer can write s_2 with a weight w (combine over all paths that write s_2)
  - $\tau(s_1, s_2) = w$

Thread Summarization

For:	Construct:
$(g, l_i) \rightarrow_{Ti}^* (g', l_i')$	τ_i [TACAS'08]
$(g, l_1, \ldots, l_i, \ldots, l_n) \Rightarrow_{Ti}^* (g', l_1, \ldots, l_i', \ldots, l_n)$	τ_i^e
$\Rightarrow^c$ equals $\Rightarrow_{T1}^* \cup \ldots \cup \Rightarrow_{Tn}^*$	$\tau_c = \tau_1^e \cup \tau_2^e \cup \ldots \cup \tau_n^e$
$\Rightarrow^c \ldots \Rightarrow^c \Rightarrow^c$	$\tau_c ; \ldots ; \tau_c ; \tau_c$

How Thread Summarization Works

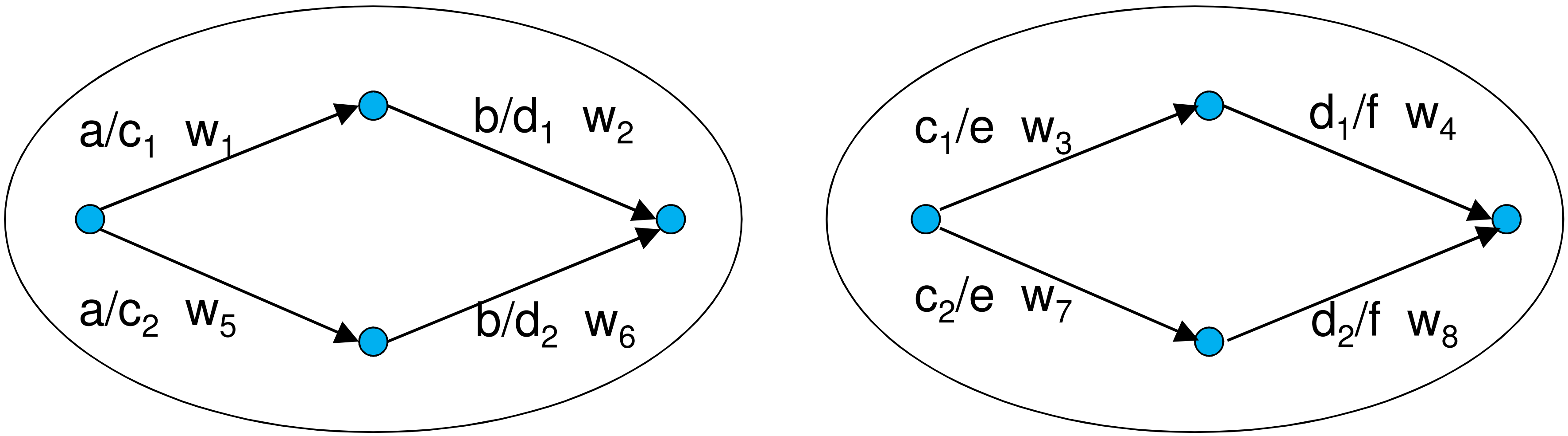
- For a single thread:
  - $\tau_i(s_1,s_2)$ = Reachable(s_1,s_2)
  - $\tau_i(s_1,s_2)$ = MOP(s_1,s_2)
- Definition of composition
  - $\tau_3(s_1,s_2) = \vee_s \{ \tau_1(s_1,s) \wedge \tau_2(s,s_2) \}$
  - $\tau_3(s_1,s_2) = \oplus_s \{ \tau_1(s_1,s) \otimes \tau_2(s,s_2) \}$
- Consider the path:
  - $\underbrace{(g_1,l_1,l_2)}_{s_1} \rightarrow_{T1}^* \underbrace{(g_2,l_1',l_2)}_{s} \rightarrow_{T2}^* \underbrace{(g_3,l_1',l_2')}_{s_2}$

$$\text{MOP}(s_1,s_2) = \oplus_s \quad \underbrace{\tau_1(s_1,s)}_{s_1 \ldots s} \quad \otimes \quad \underbrace{\tau_2(s, s_2)}_{s \ldots s_2}$$

$\tau_3(s_1,s_2)$

Composing Transducers

- $\tau_3(s_1,s_2) = \oplus_s \{ \tau_1(s_1,s) \otimes \tau_2(s,s_2) \}$



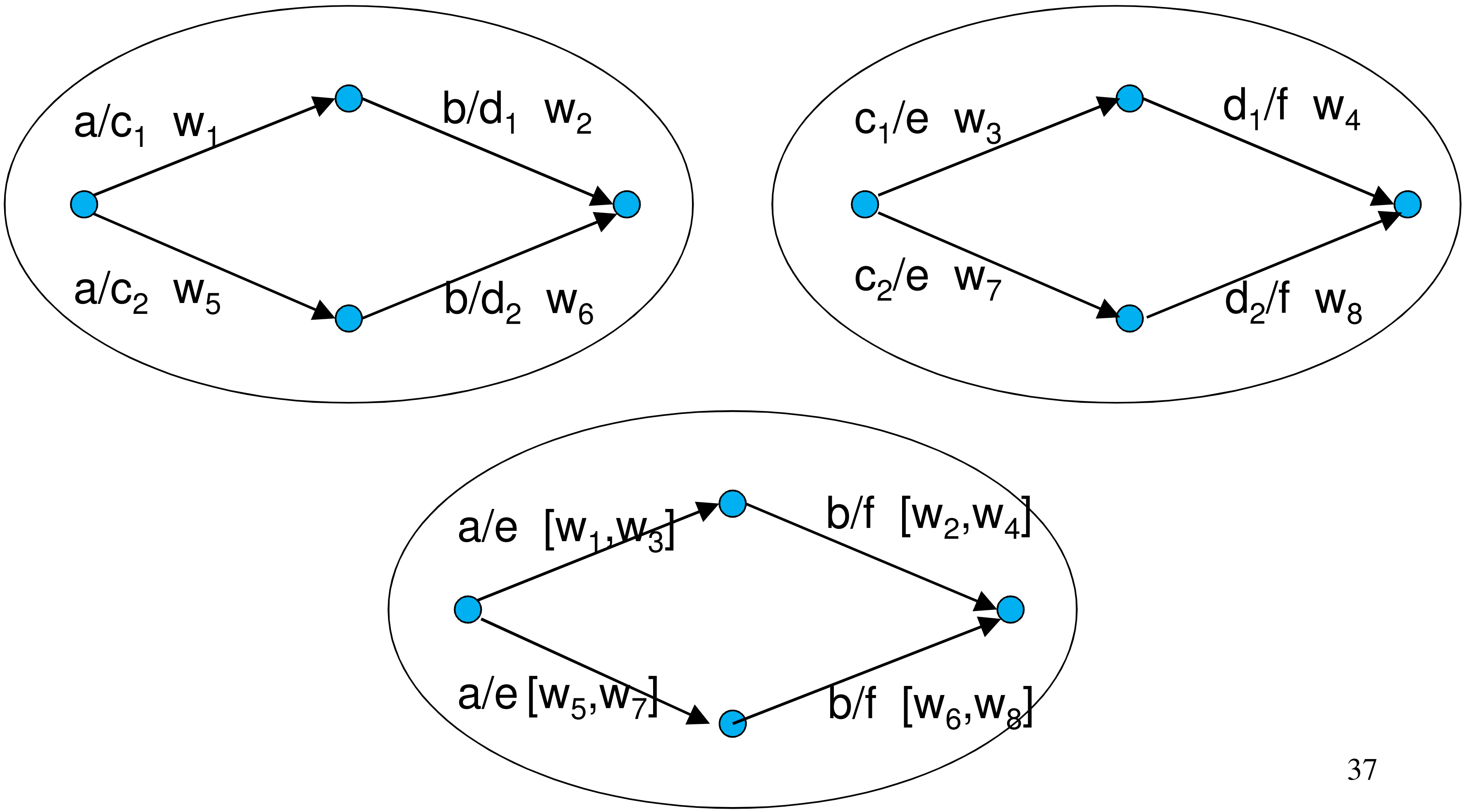
$ab \rightarrow c_1d_1 \quad w_1 \otimes w_2$
$ab \rightarrow c_2d_2 \quad w_5 \otimes w_6$

$c_1d_1 \rightarrow ef \quad w_3 \otimes w_4$
$c_2d_2 \rightarrow ef \quad w_7 \otimes w_8$

$$ab \rightarrow ef \quad w_1 \otimes w_2 \otimes w_3 \otimes w_4 \oplus w_5 \otimes w_6 \otimes w_7 \otimes w_8$$

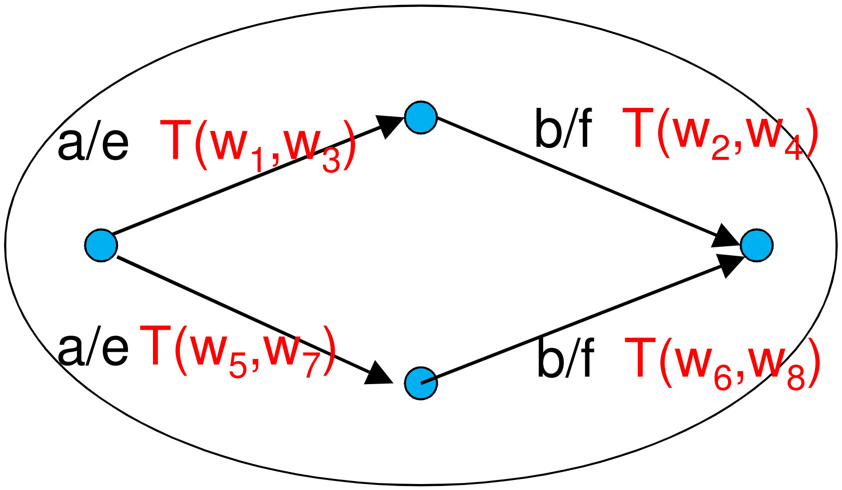
Composing Transducers

- $\tau_3(s_1,s_2) = \oplus_s \{ \tau_1(s_1,s) \otimes \tau_2(s,s_2) \}$



Composing Transducers

- $\tau_3(s_1,s_2) = \oplus_s \{ \tau_1(s_1,s) \otimes \tau_2(s,s_2) \}$



$$T(w_1 \otimes w_2, w_3 \otimes w_4)$$
$$\oplus\ T(w_5 \otimes w_6, w_7 \otimes w_8)$$

DeTensor

$$w_1 \otimes w_2 \otimes w_3 \otimes w_4$$
$$\oplus\ w_5 \otimes w_6 \otimes w_7 \otimes w_8$$

Summary

- We gave an algorithm for intersecting weighted automata
  - Extend need not be commutative
  - Requires tensor product for “shuffling”
- Generalizes to transducer composition
- Solves Context-Bounded Analysis