Verification and Synthesis of Parameterized Concurrent Systems

Anirban Majumdar

Supervised by: Patricia Bouyer, Nathalie Bertrand

September 30, 2021

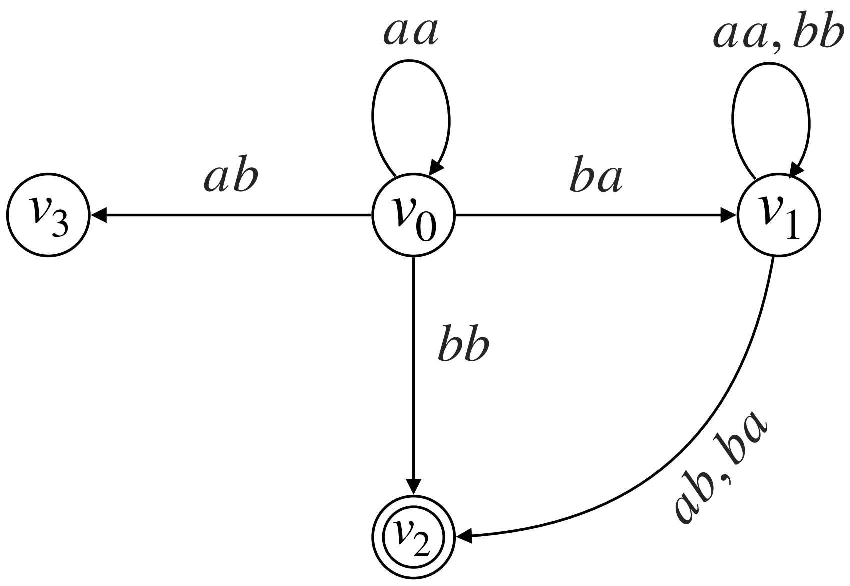
université PARIS-SACLAY

Inria INVENTEURS DU MONDE NUMÉRIQUE

Part - II

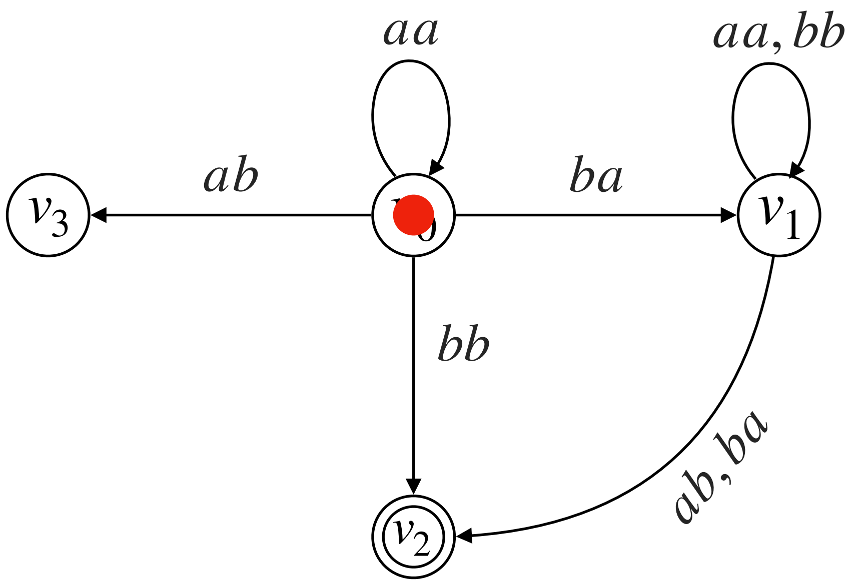
Parameterized Concurrent Games

2-player Concurrent games on graphs



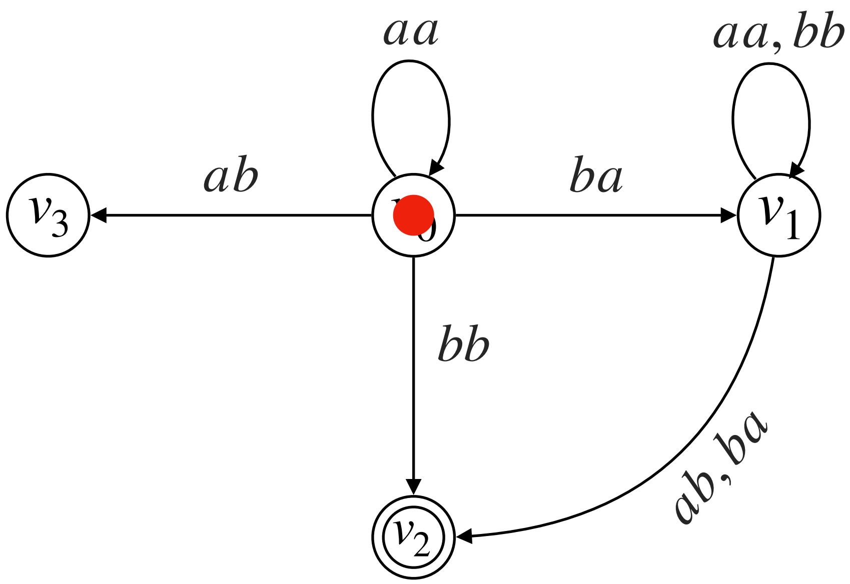
- Finite set of actions: $\Sigma = \{a, b\}$.
- The game proceeds as follows:

2-player Concurrent games on graphs



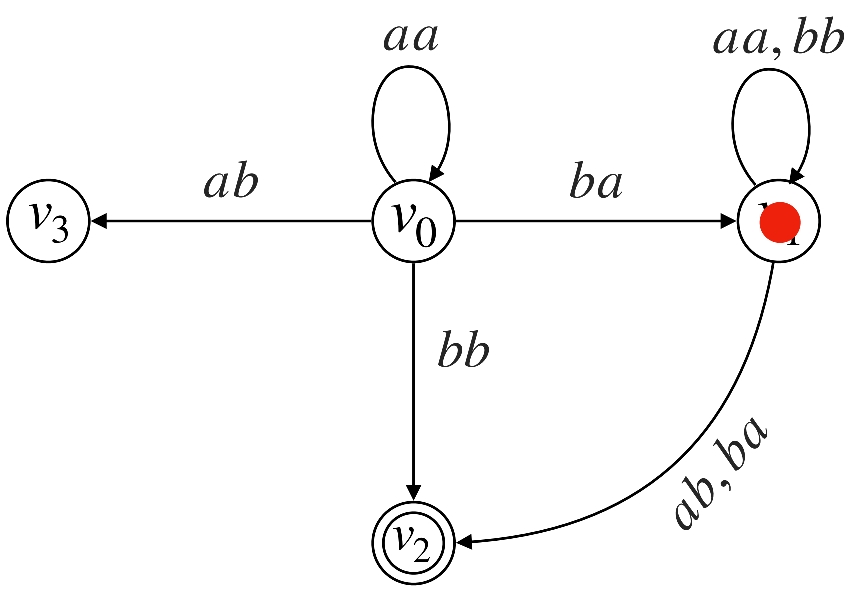
- Finite set of actions: $\Sigma = \{a, b\}$.
- The game proceeds as follows:
  - Game starts at initial vertex.

2-player Concurrent games on graphs



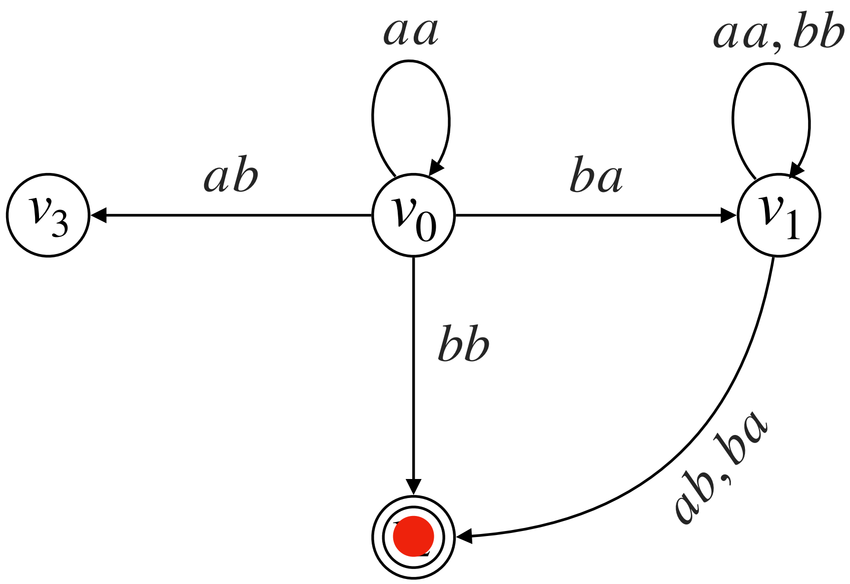
- Finite set of actions: $\Sigma = \{a, b\}$.
- The game proceeds as follows:
  - Game starts at initial vertex.
  - Players choose actions simultaneously.
  - Next vertex is determined by the chosen actions.

2-player Concurrent games on graphs



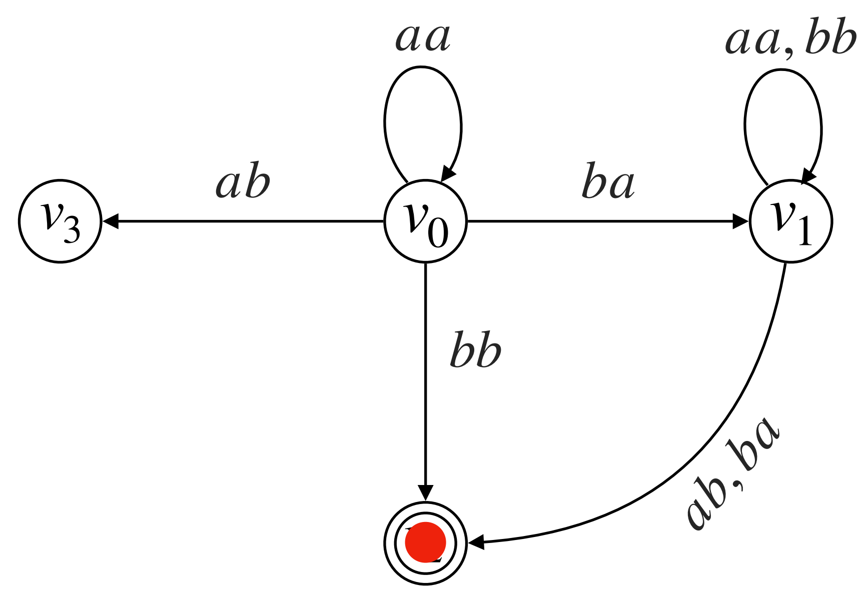
- Finite set of actions: $\Sigma = \{a, b\}$.
- The game proceeds as follows:
  - Game starts at initial vertex.
  - Players choose actions simultaneously.
  - Next vertex is determined by the chosen actions.

2-player Concurrent games on graphs



- Finite set of actions: $\Sigma = \{a, b\}$.
- The game proceeds as follows:
    - Game starts at initial vertex.
    - Players choose actions simultaneously.
    - Next vertex is determined by the chosen actions.

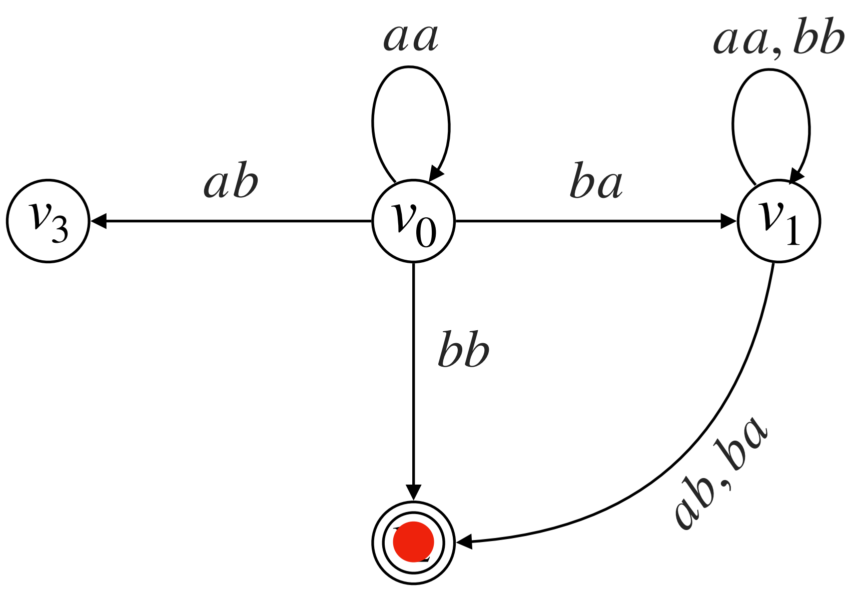
2-player Concurrent games on graphs



- Finite set of actions: $\Sigma = \{a, b\}$.
- The game proceeds as follows:
  - Game starts at initial vertex.
  - Players choose actions simultaneously.
  - Next vertex is determined by the chosen actions.

Player 1 needs to win against all strategies of player 2.

2-player Concurrent games on graphs



- Finite set of actions: $\Sigma = \{a, b\}$.
- The game proceeds as follows:
  - Game starts at initial vertex.
  - Players choose actions simultaneously.
  - Next vertex is determined by the chosen actions.

Player 1 needs to win against all strategies of player 2.

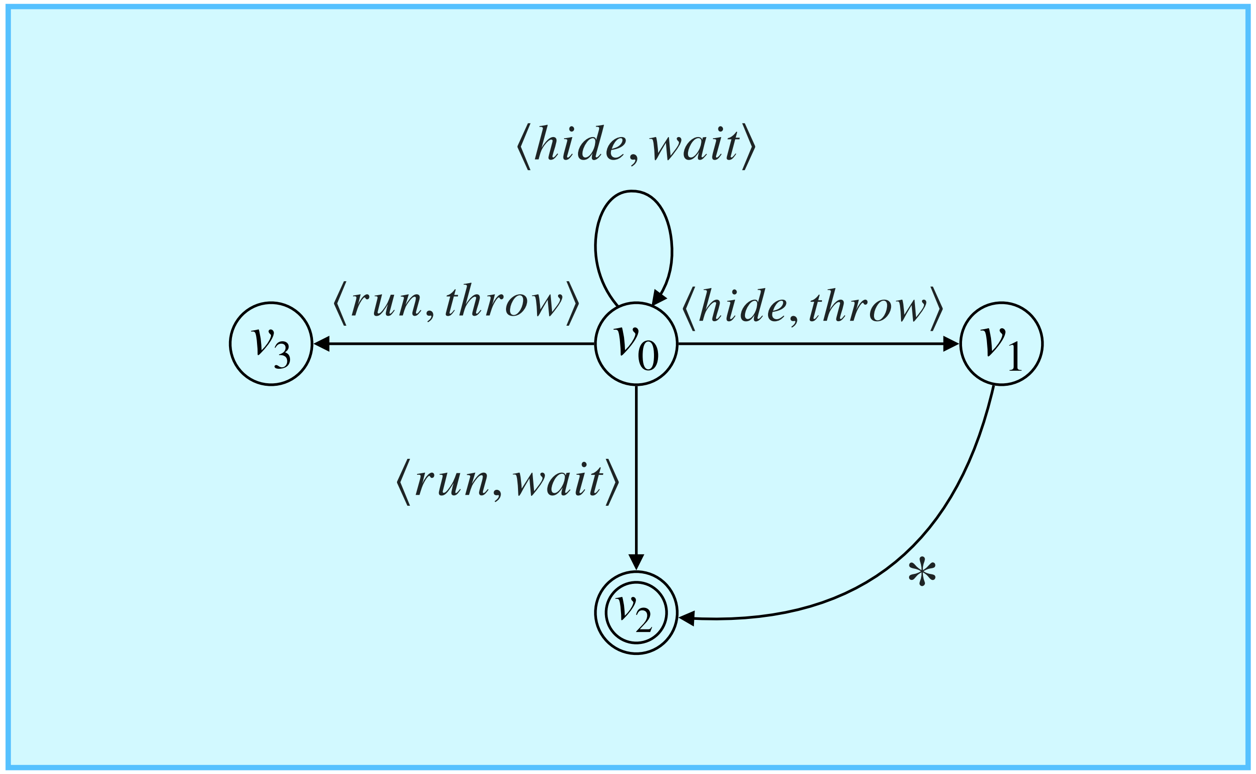
- Examples of winning objectives: Reachability, Safety…

Example

[Alfaro, Henzinger, Kupferman '07]

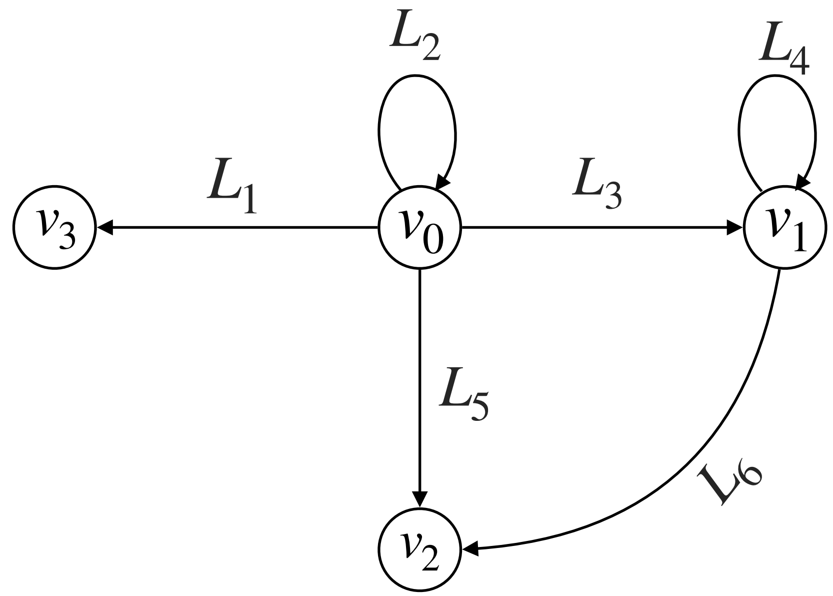
Hide-or-run example

- Player 1 wants to reach home safely when Player 2 wants to throw a snowball at him.



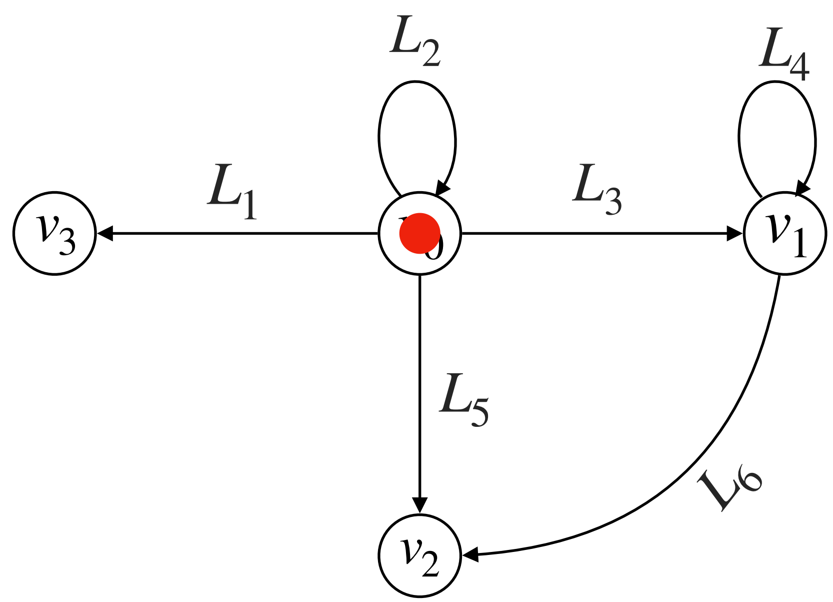
- No player has a winning strategy.

Parameterized concurrent game arena



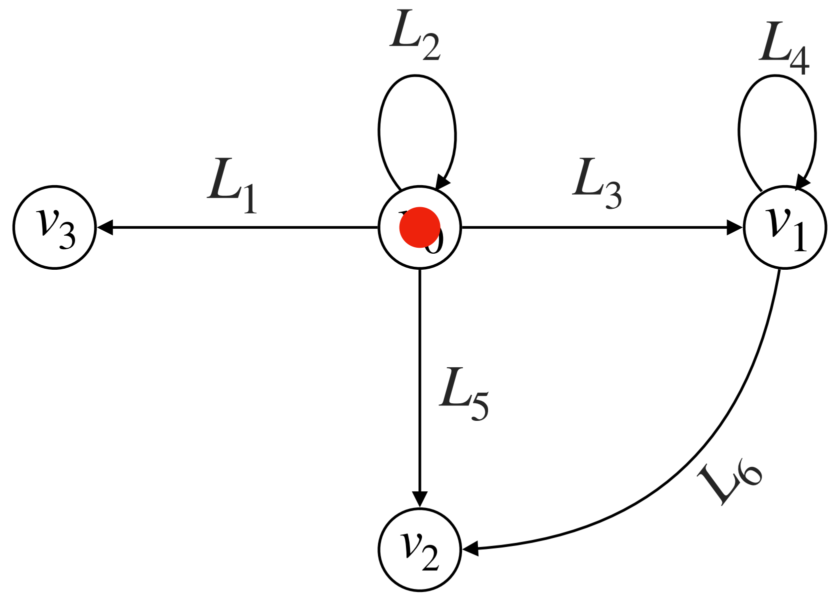
- Finite set of actions: Σ .
- $L_i \subseteq \Sigma^*$.
- Number of players is unknown.
- The game proceeds as follows:

Parameterized concurrent game arena



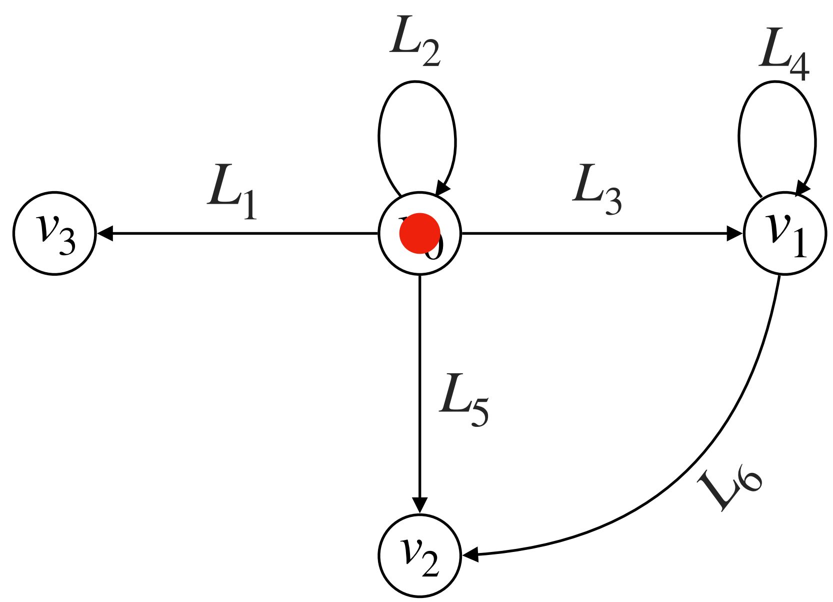
- Finite set of actions: Σ .
- $L_i \subseteq \Sigma^*$.
- Number of players is unknown.
- The game proceeds as follows:
    - Game starts at initial vertex.

Parameterized concurrent game arena



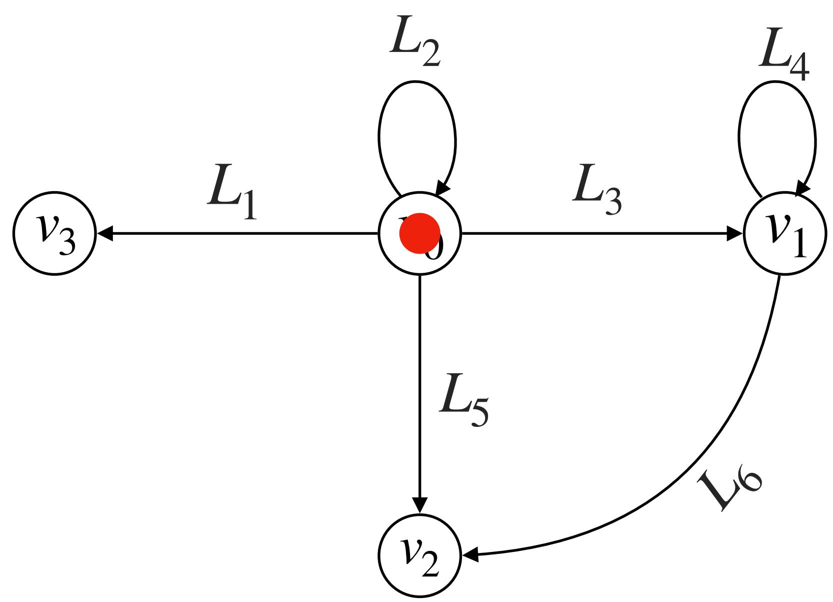
- Finite set of actions: Σ .
- $L_i \subseteq \Sigma^*$.
- Number of players is unknown.
- The game proceeds as follows:
    - Game starts at initial vertex.
    - Adversary fixes k - the number of players (unknown to players).

Parameterized concurrent game arena



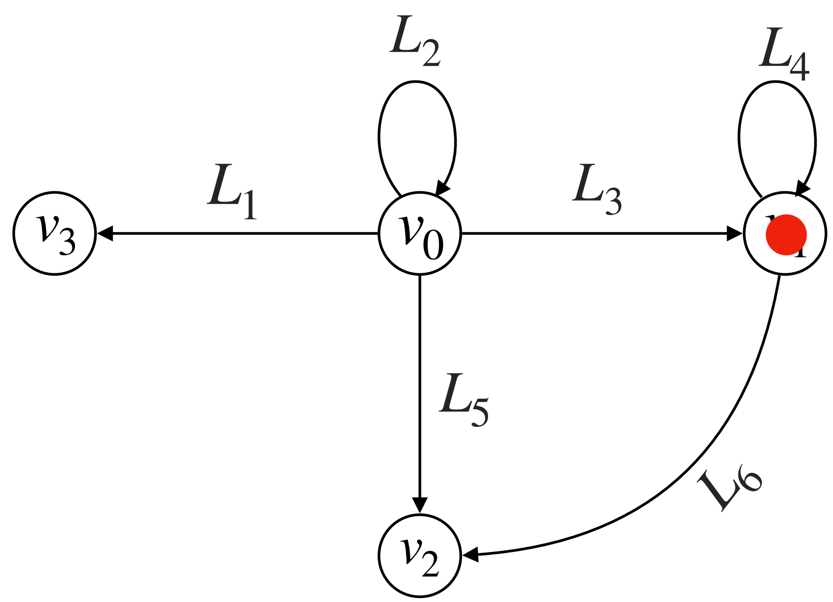
- Finite set of actions: Σ .
- $L_i \subseteq \Sigma^*$.
- Number of players is unknown.
- The game proceeds as follows:
    - Game starts at initial vertex.
    - Adversary fixes k - the number of players (unknown to players).
    - Players choose actions simultaneously: they form a word $w = a_1 \ a_2 \ \ldots \ a_k$.

Parameterized concurrent game arena



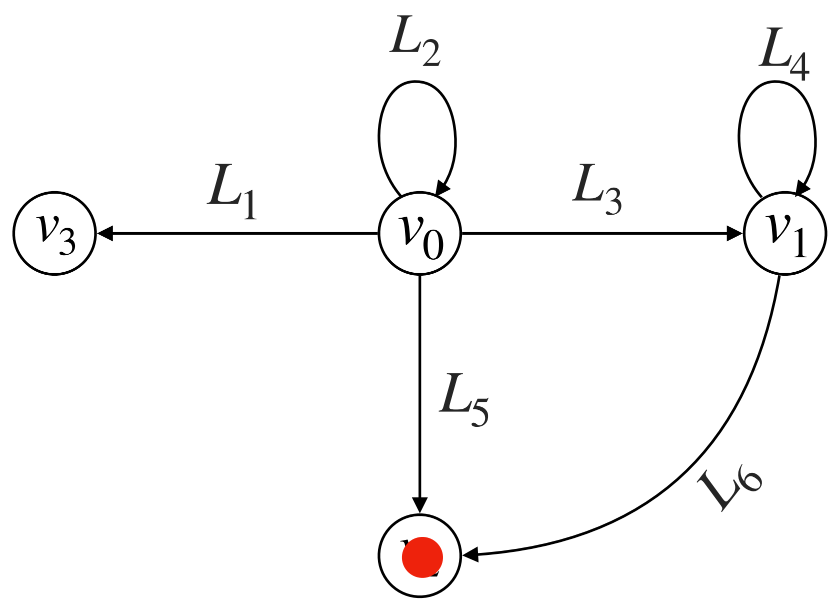
- Finite set of actions: Σ .
- $L_i \subseteq \Sigma^*$.
- Number of players is unknown.
- The game proceeds as follows:
    - Game starts at initial vertex.
    - Adversary fixes k - the number of players (unknown to players).
    - Players choose actions simultaneously: they form a word $w = a_1 \ a_2 \ \ldots \ a_k$.
    - Next vertex is such that $w \in L_i$ (non-determinism is resolved by adversary).

Parameterized concurrent game arena



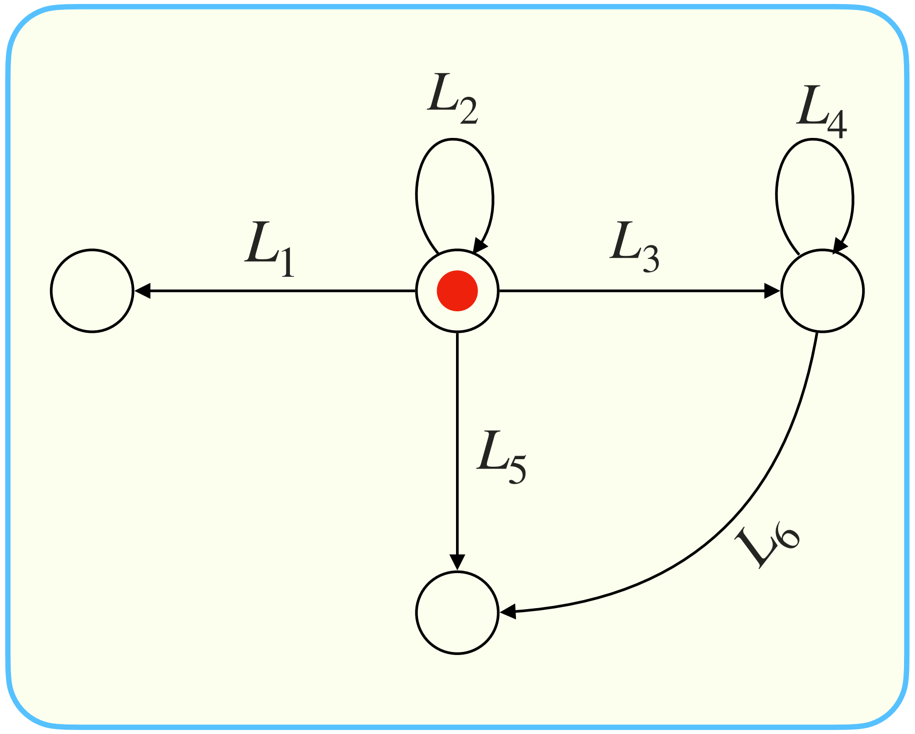
- Finite set of actions: Σ .
- $L_i \subseteq \Sigma^*$.
- Number of players is unknown.
- The game proceeds as follows:
  - Game starts at initial vertex.
  - Adversary fixes k - the number of players (unknown to players).
  - Players choose actions simultaneously: they form a word $w = a_1 \; a_2 \; \ldots \; a_k$.
  - Next vertex is such that $w \in L_i$ (non-determinism is resolved by adversary).

Parameterized concurrent game arena



- Finite set of actions: Σ .
- $L_i \subseteq \Sigma^*$.
- Number of players is unknown.
- The game proceeds as follows:
  - Game starts at initial vertex.
  - Adversary fixes k - the number of players (unknown to players).
  - Players choose actions simultaneously: they form a word $w = a_1\ a_2\ \ldots\ a_k$.
  - Next vertex is such that $w \in L_i$ (non-determinism is resolved by adversary).

Decision problems

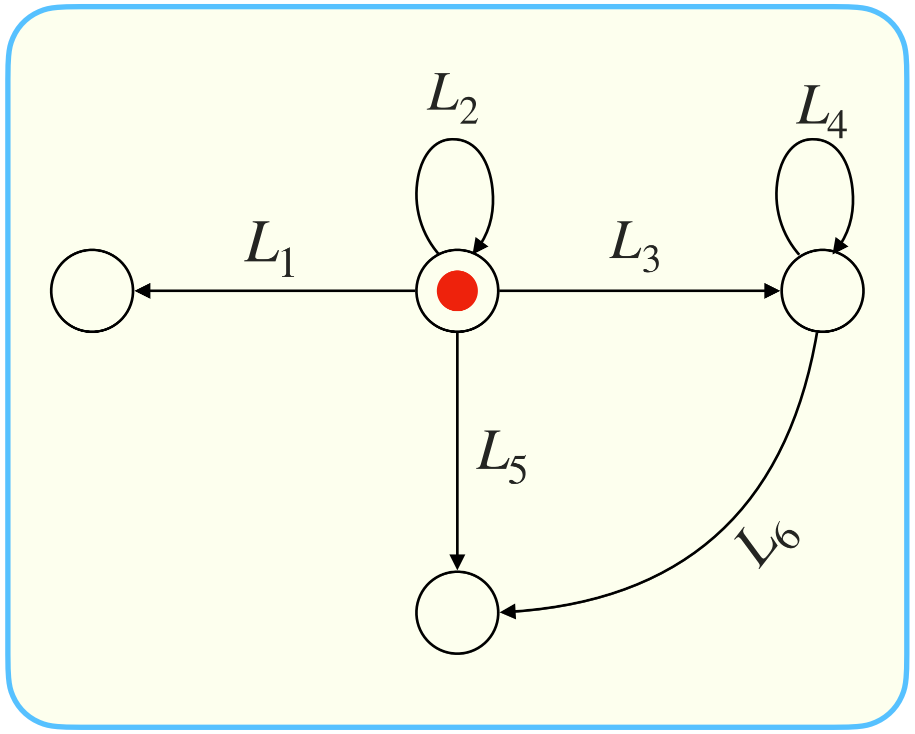


A distinguished player trying to achieve a goal against arbitrary number of opponents.

(Example: Server-clients)

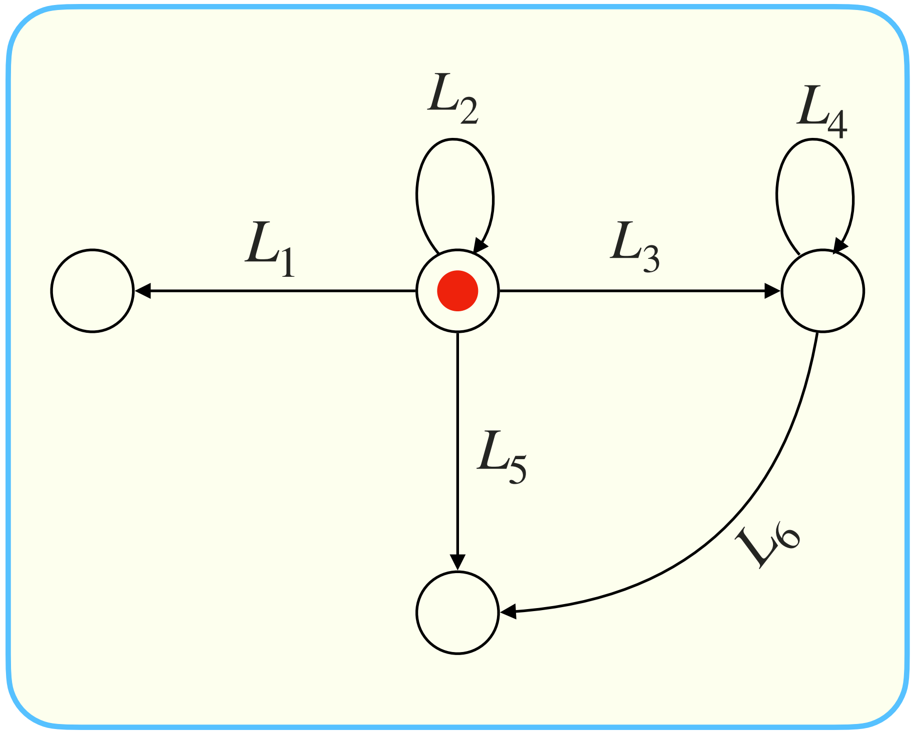
(Example: Fleet of drones)

Decision problems



A distinguished player trying to achieve a goal against arbitrary number of opponents.

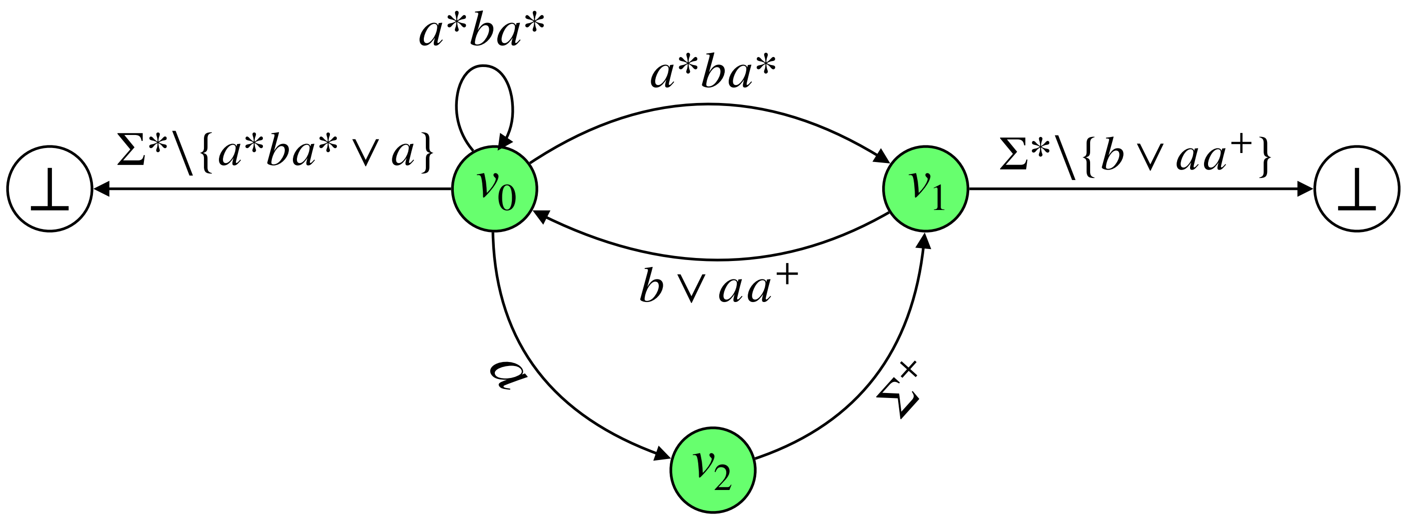
(Example: Server-clients)



Arbitrary number of players trying to achieve a common goal as a coalition.

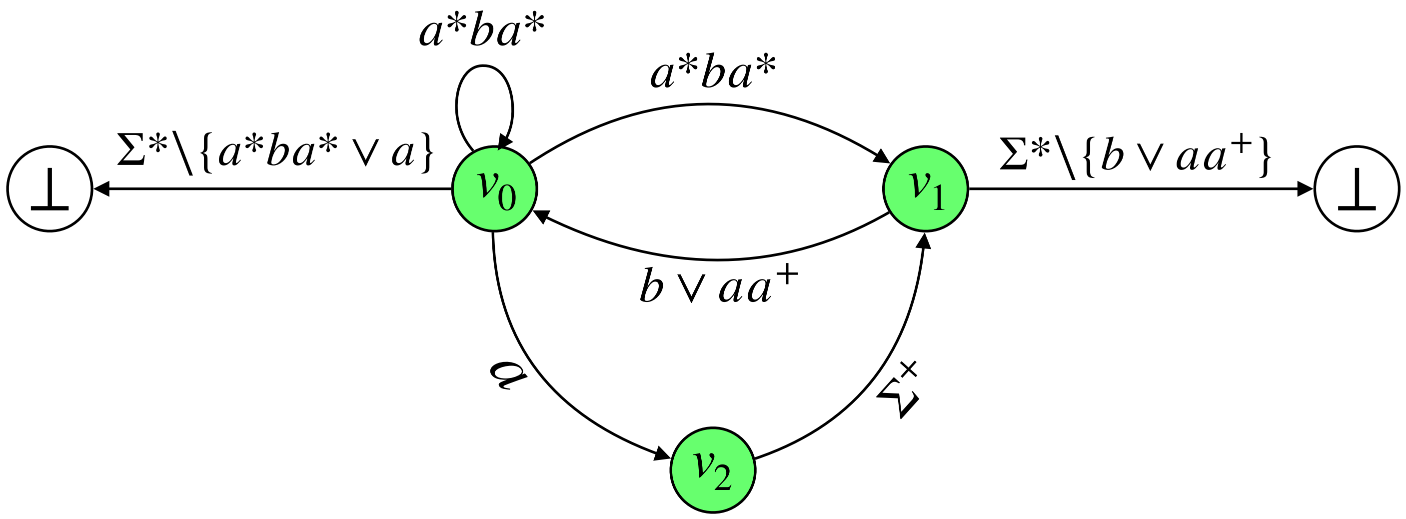
(Example: Fleet of drones)

Safe coalition problem



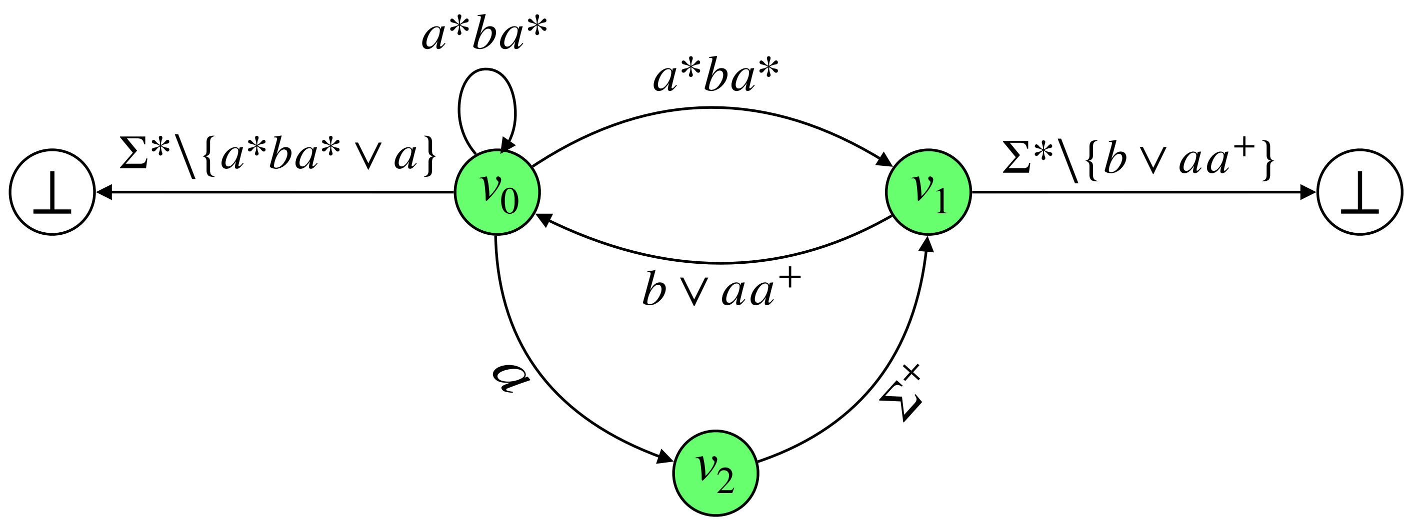
- **Strategy** of player i is $\sigma_i : V^+ \rightarrow \Sigma$.

Safe coalition problem



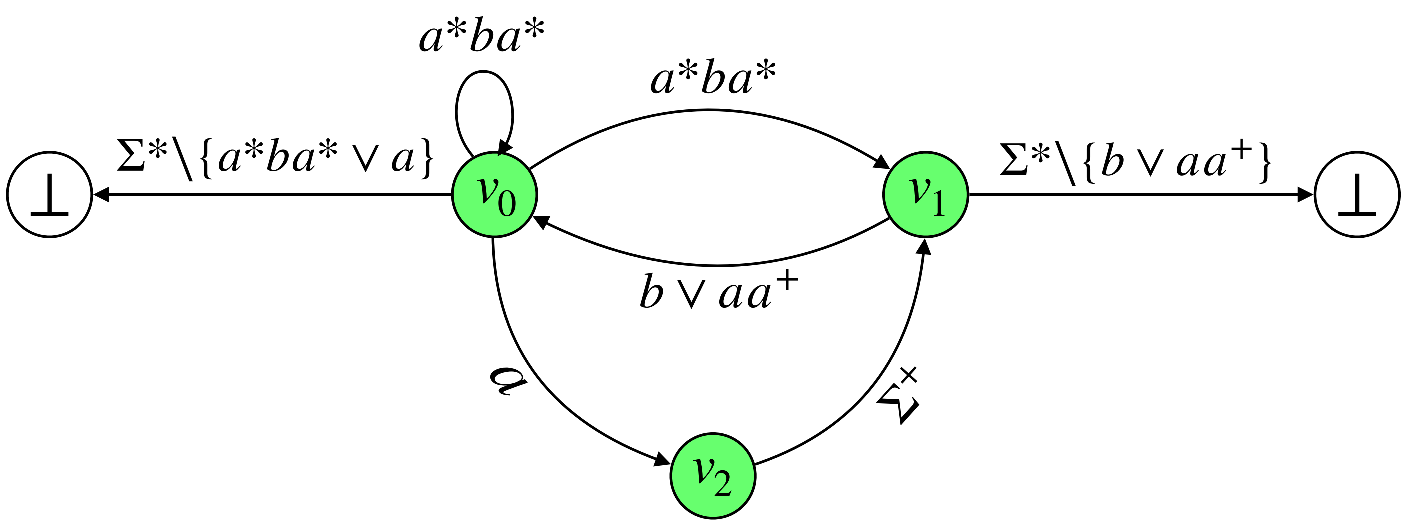
- Strategy of player i is $\sigma_i : V^+ \to \Sigma$.
- A coalition strategy is $\widetilde{\sigma} = \langle \sigma_1, \sigma_2, \ldots \rangle$. Equivalently, $\widetilde{\sigma} : V^+ \to \Sigma^\omega$.

Safe coalition problem



- Strategy of player i is $\sigma_i : V^+ \to \Sigma$.
- A coalition strategy is $\widetilde{\sigma} = \langle \sigma_1, \sigma_2, \ldots \rangle$. Equivalently, $\widetilde{\sigma} : V^+ \to \Sigma^\omega$.
- $Out^k(v_0, \widetilde{\sigma})$ = set of plays induced by $\widetilde{\sigma}$ from v_0 with k players.

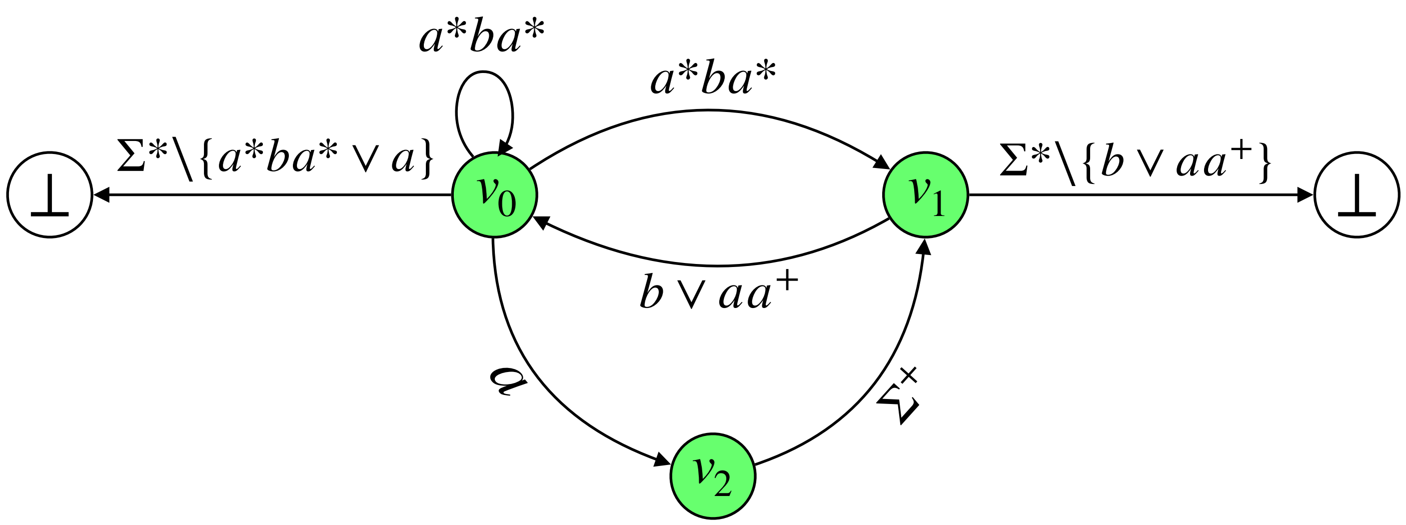
Safe coalition problem



- Strategy of player i is $\sigma_i : V^+ \to \Sigma$.
- A coalition strategy is $\widetilde{\sigma} = \langle \sigma_1, \sigma_2, \ldots \rangle$. Equivalently, $\widetilde{\sigma} : V^+ \to \Sigma^\omega$.
- $Out^k(v_0, \widetilde{\sigma})$ = set of plays induced by $\widetilde{\sigma}$ from v_0 with k players.

The coalition wins if they can keep the play within a safe set of vertices.

Safe coalition problem



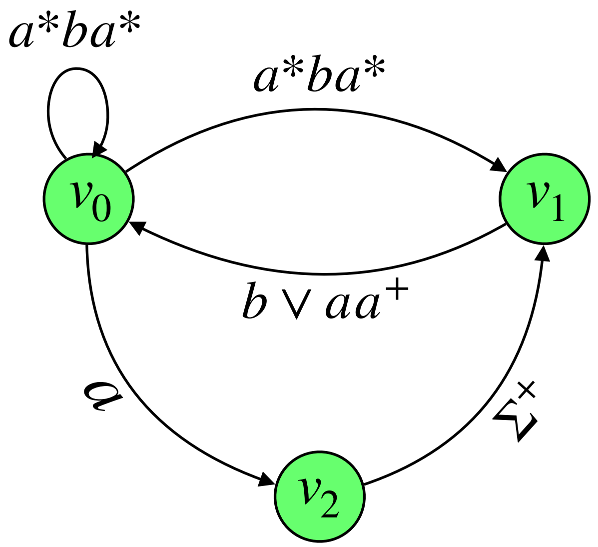
- Strategy of player i is $\sigma_i : V^+ \to \Sigma$.
- A coalition strategy is $\widetilde{\sigma} = \langle \sigma_1, \sigma_2, \ldots \rangle$. Equivalently, $\widetilde{\sigma} : V^+ \to \Sigma^\omega$.
- $Out^k(v_0, \widetilde{\sigma})$ = set of plays induced by $\widetilde{\sigma}$ from v_0 with k players.

The coalition wins if they can keep the play within a safe set of vertices.

Input: Arena $\mathscr{A}$, initial vertex $v_0 \in V$ and set of safe vertices S.

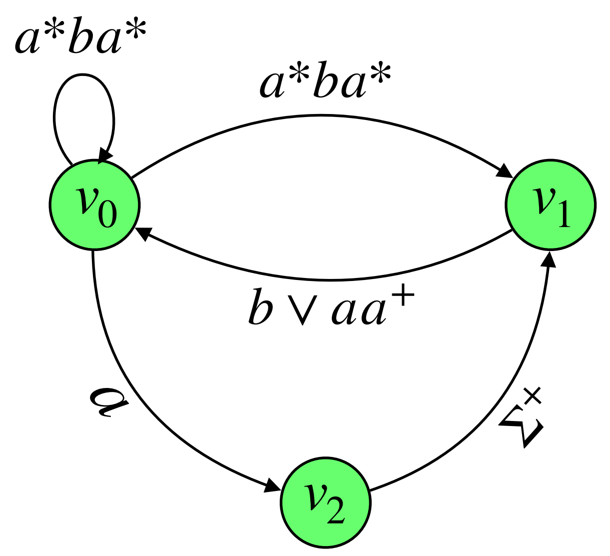
Output: Yes iff $\exists \widetilde{\sigma} . \forall k . Out^k(v_0, \widetilde{\sigma}) \subseteq S^\omega$.

Example



- $\Sigma = \{a, b\}$.
- Unspecified transitions lead to a losing vertex $\perp$.
- Coalition needs to stay within the safe vertices.

Example

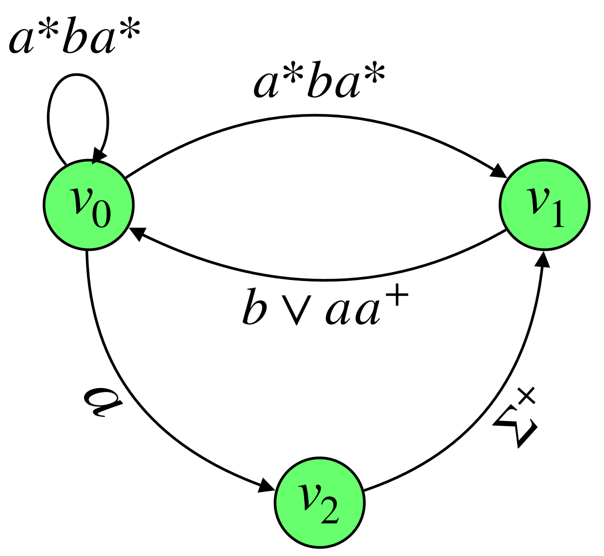


- $\Sigma = \{a, b\}$.
- Unspecified transitions lead to a losing vertex $\perp$.
- Coalition needs to stay within the safe vertices.

▷ A coalition winning strategy:

$$\widetilde{\sigma}(v_0) = aba^\omega;\ \widetilde{\sigma}(v_0v_2) = a^\omega;$$
$$\widetilde{\sigma}(v_0v_1) = a^\omega;\ \widetilde{\sigma}(v_0v_2v_1) = b^\omega\ .$$

Example



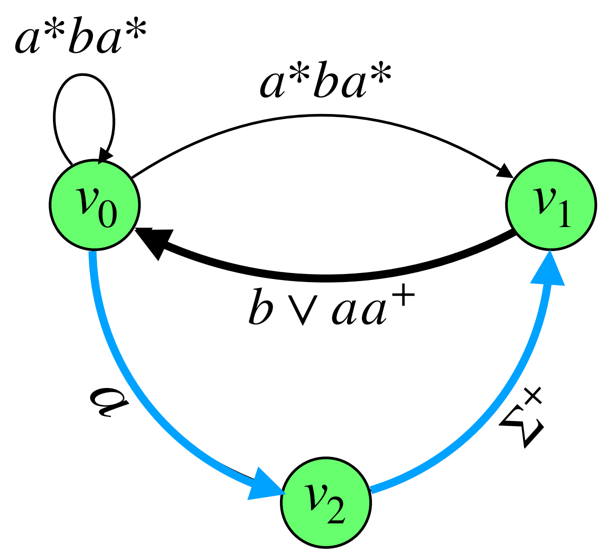
- $\Sigma = \{a, b\}$.
- Unspecified transitions lead to a losing vertex $\perp$.
- Coalition needs to stay within the safe vertices.

A coalition winning strategy:

$$\widetilde{\sigma}(v_0) = aba^\omega;\ \widetilde{\sigma}(v_0v_2) = a^\omega;$$
$$\widetilde{\sigma}(v_0v_1) = a^\omega;\ \widetilde{\sigma}(v_0v_2v_1) = b^\omega .$$

- At v_0, coalition plays aba^ω, since any other choice leads to $\perp$ for some k.

Example



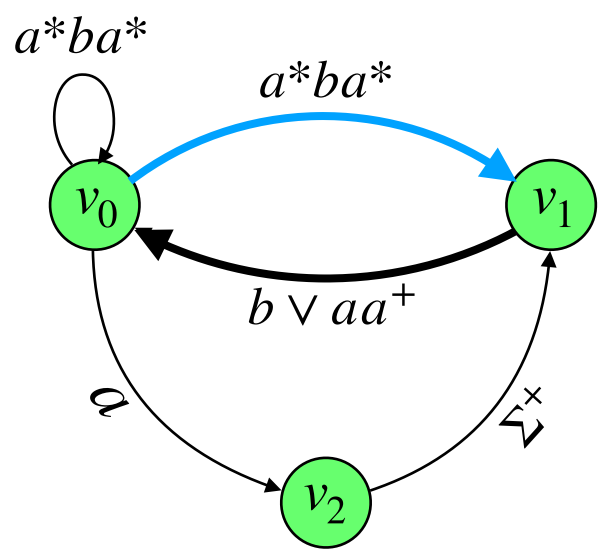
- $\Sigma = \{a, b\}$.
- Unspecified transitions lead to a losing vertex $\perp$.
- Coalition needs to stay within the safe vertices.

▶ A coalition winning strategy:

$$\widetilde{\sigma}(v_0) = aba^\omega;\ \widetilde{\sigma}(v_0v_2) = a^\omega;$$
$$\widetilde{\sigma}(v_0v_1) = a^\omega;\ \widetilde{\sigma}(v_0v_2v_1) = b^\omega .$$

- At v_0, coalition plays aba^ω, since any other choice leads to $\perp$ for some k.
- At v_1, if the history is $v_0v_2v_1$, the coalition infer there is only 1 player, hence they choose b^ω .

Example



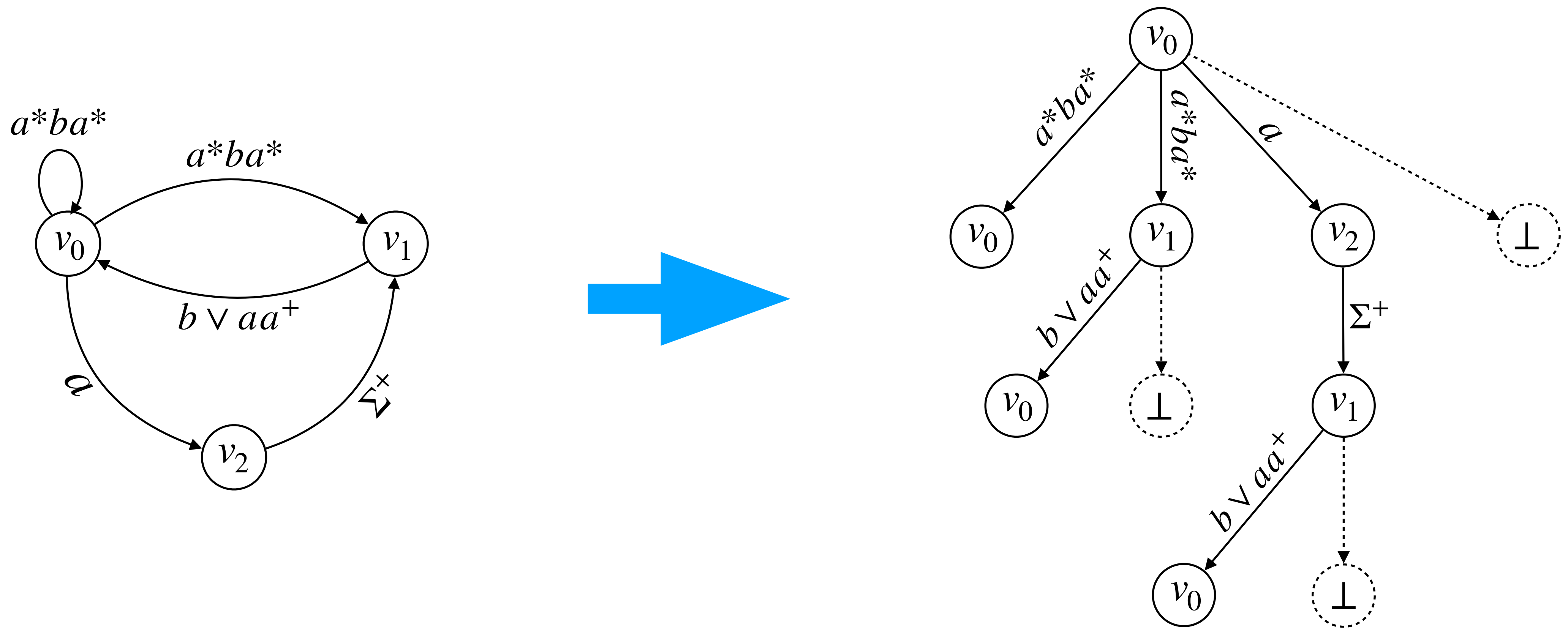
- $\Sigma = \{a, b\}$.
- Unspecified transitions lead to a losing vertex $\bot$.
- Coalition needs to stay within the safe vertices.

▶ A coalition winning strategy:

$$\widetilde{\sigma}(v_0) = aba^\omega;\ \widetilde{\sigma}(v_0v_2) = a^\omega;$$
$$\widetilde{\sigma}(v_0v_1) = a^\omega;\ \widetilde{\sigma}(v_0v_2v_1) = b^\omega .$$

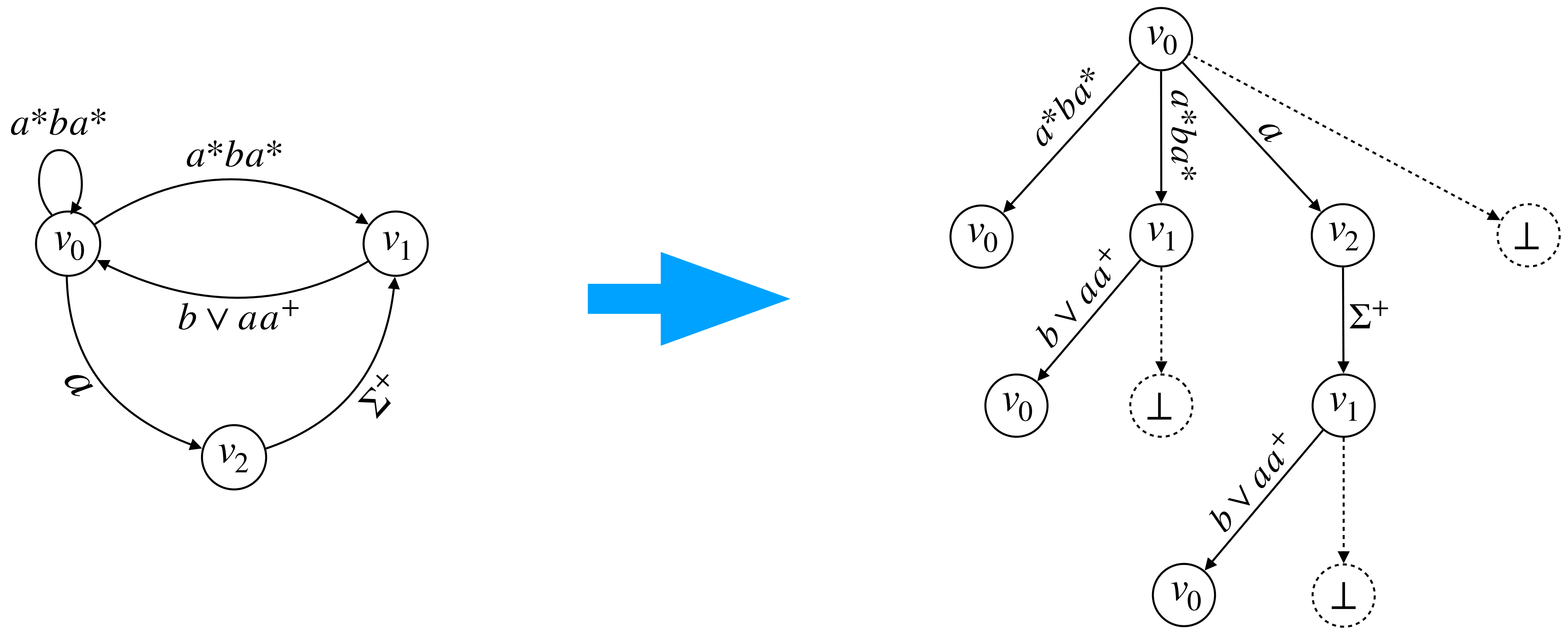
- At v_0, coalition plays aba^ω, since any other choice leads to $\bot$ for some k.
- At v_1, if the history is $v_0v_2v_1$, the coalition infer there is only 1 player, hence they choose b^ω .
- At v_1, if the history is v_0v_1, coalition infer there is at least 2 players, hence they choose a^ω .

Resolution of safe coalition problem



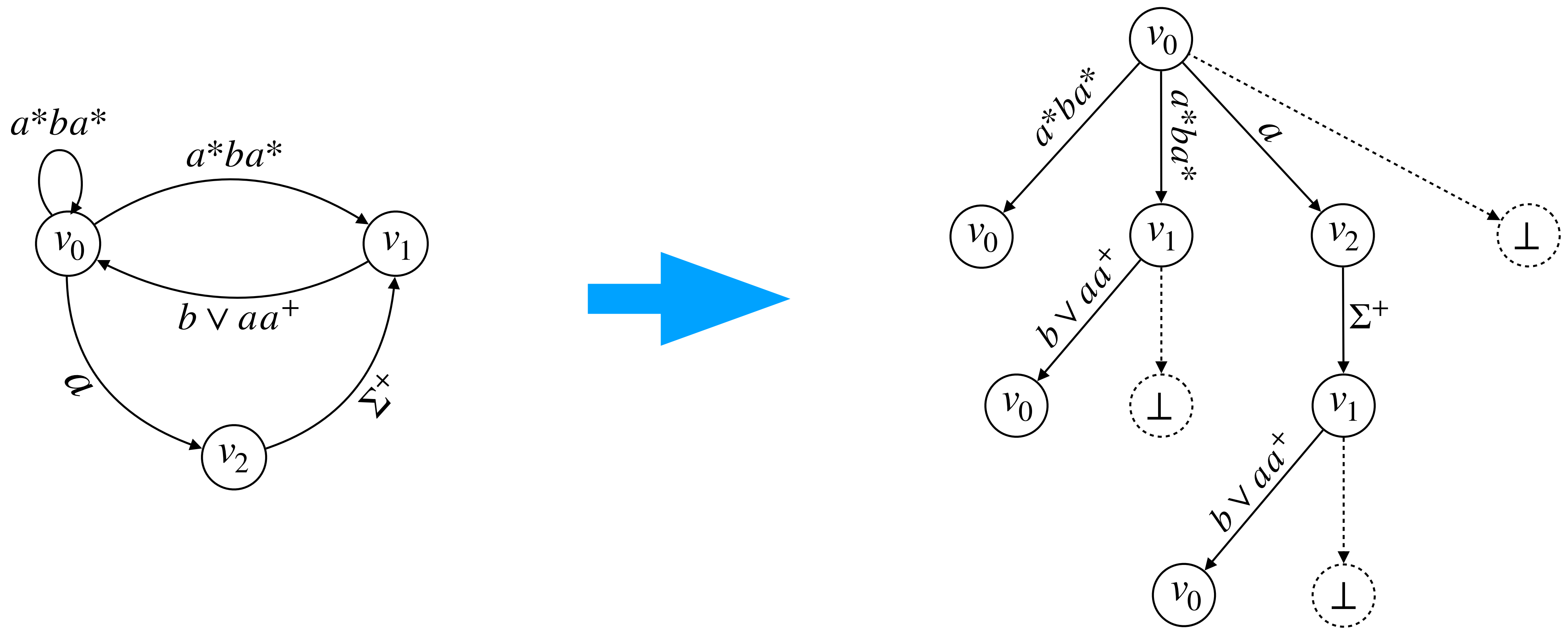
- Unfold arena $\mathscr{A}$ to a finite tree.
    - Label nodes with corresponding vertices, and edges with languages.

Resolution of safe coalition problem



- Unfold arena $\mathscr{A}$ to a finite tree.
    - Label nodes with corresponding vertices, and edges with languages.
- Terminate a branch if:
    - either some label repeats in the same branch,
    - or the label is not in S.

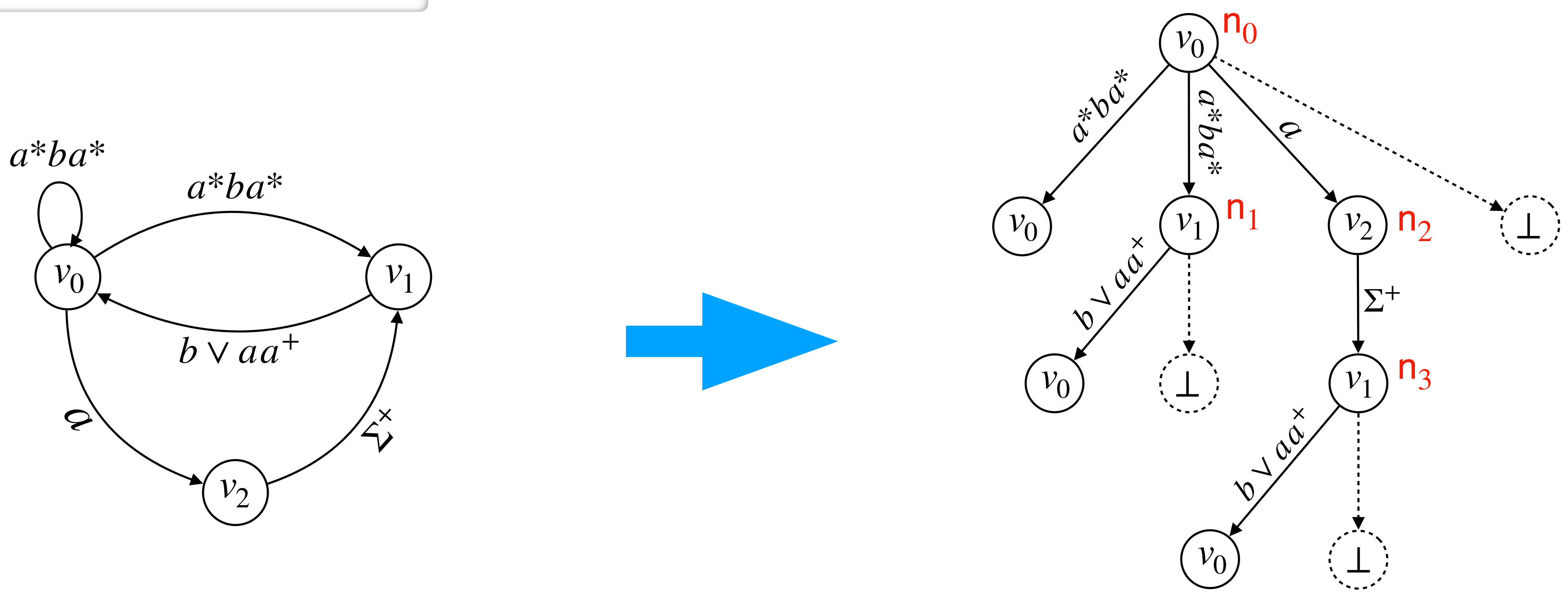
Resolution of safe coalition problem



- Unfold arena $\mathscr{A}$ to a finite tree.
  - Label nodes with corresponding vertices, and edges with languages.
- Terminate a branch if:
  - either some label repeats in the same branch,
  - or the label is not in S.
- Intuitively, if a vertex repeats in $\mathscr{A}$, coalition may take the same strategy.
  - If it ensures safety in the first occurrence, then also for the later.

Resolution of safe coalition problem

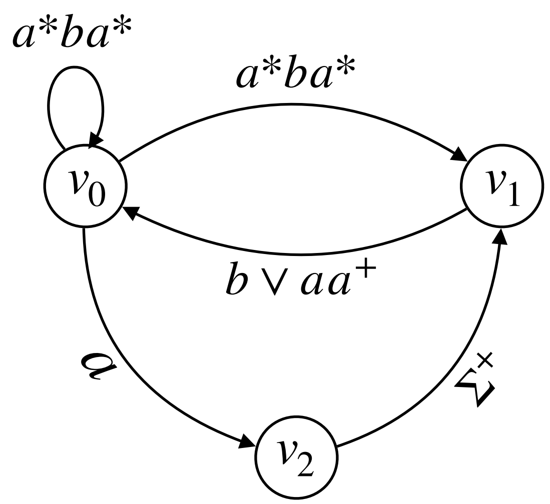
Correctness of Tree Unfolding

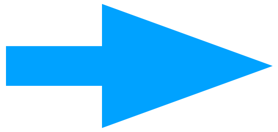


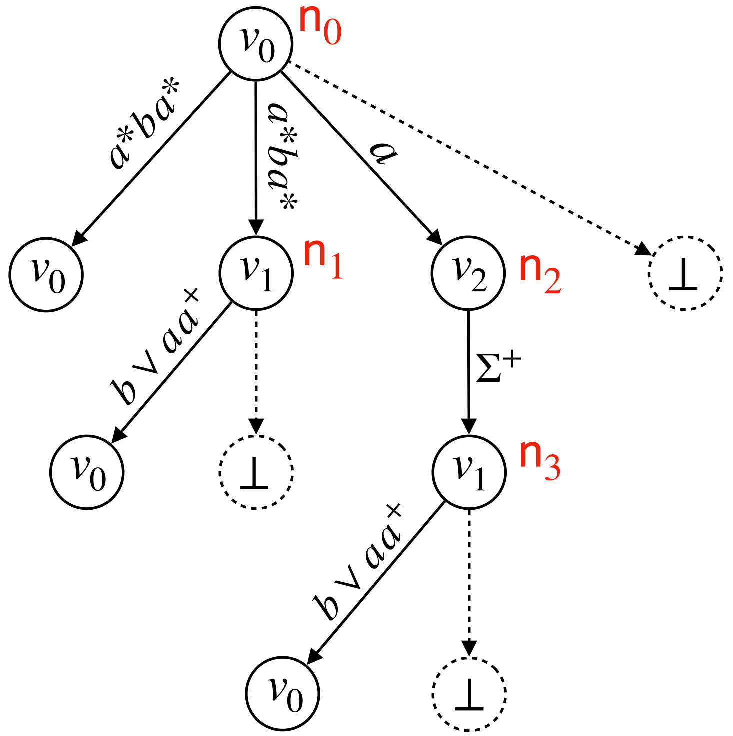
A coalition Strategy in the tree is a mapping $\tau : N_{int} \rightarrow \Sigma^\omega$.

Resolution of safe coalition problem

Correctness of Tree Unfolding





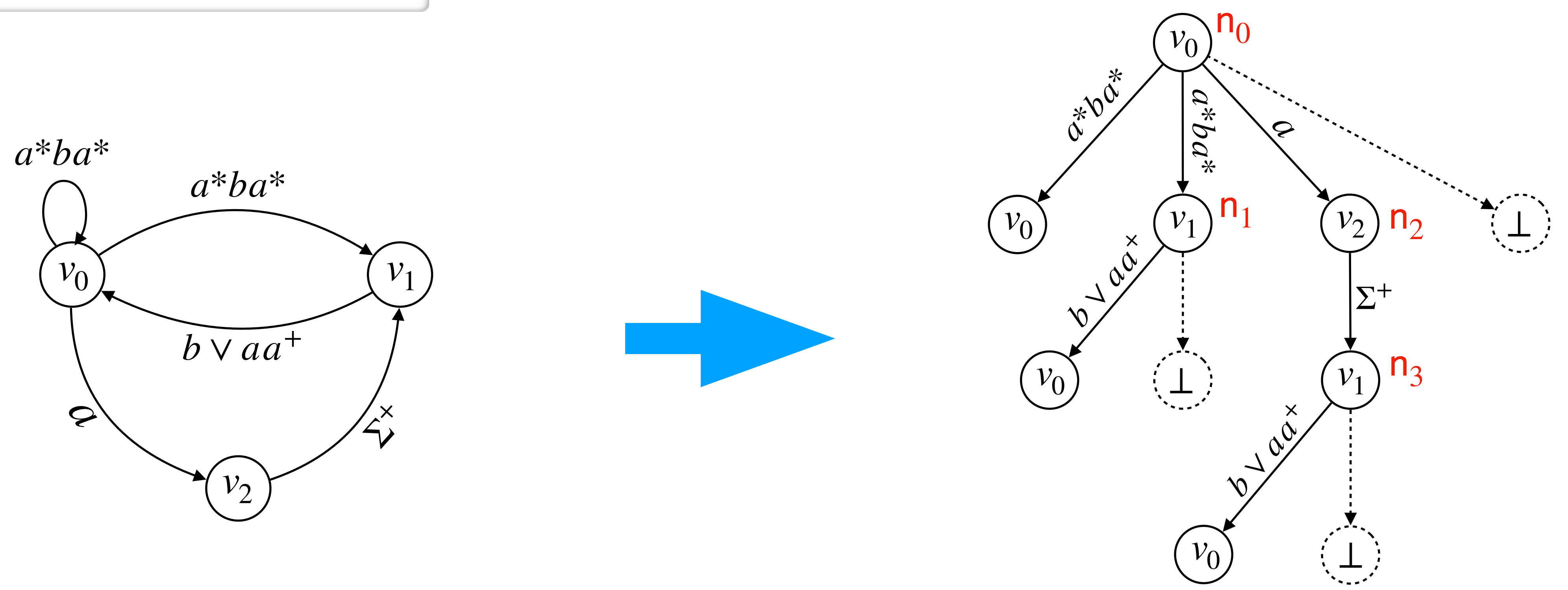


- A coalition Strategy in the tree is a mapping $\tau : N_{int} \to \Sigma^\omega$.

The coalition wins if they can reach a safe leaf.

Resolution of safe coalition problem

Correctness of Tree Unfolding



- A coalition Strategy in the tree is a mapping $\tau : N_{int} \to \Sigma^{\omega}$.

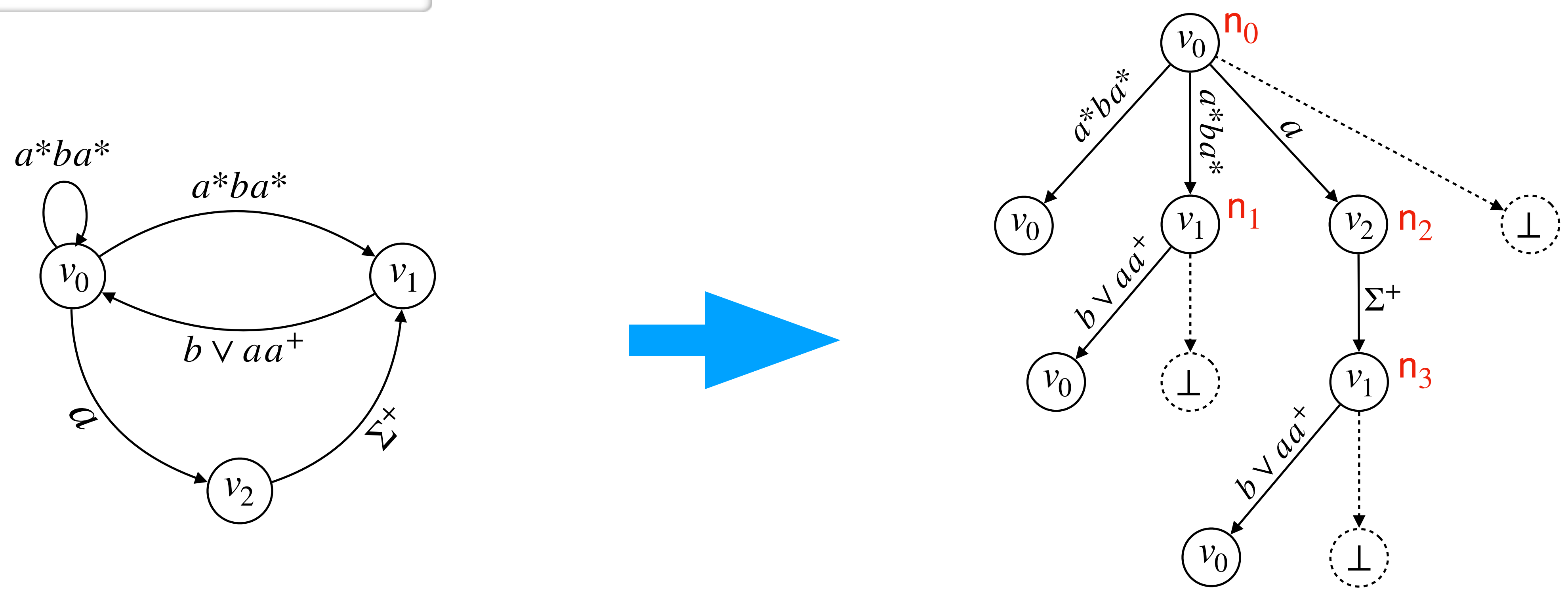
The coalition wins if they can reach a safe leaf.

Coalition has a winning strategy in $\mathscr{A}$ ⟺ Coalition has a winning strategy in $\mathscr{T}$

- Proof idea: any history V^+ in $\mathscr{A}$ uniquely maps to an internal node in $\mathscr{T}$.

Resolution of safe coalition problem

Correctness of Tree Unfolding



- A coalition Strategy in the tree is a mapping $\tau : N_{int} \rightarrow \Sigma^{\omega}$.

The coalition wins if they can reach a safe leaf.

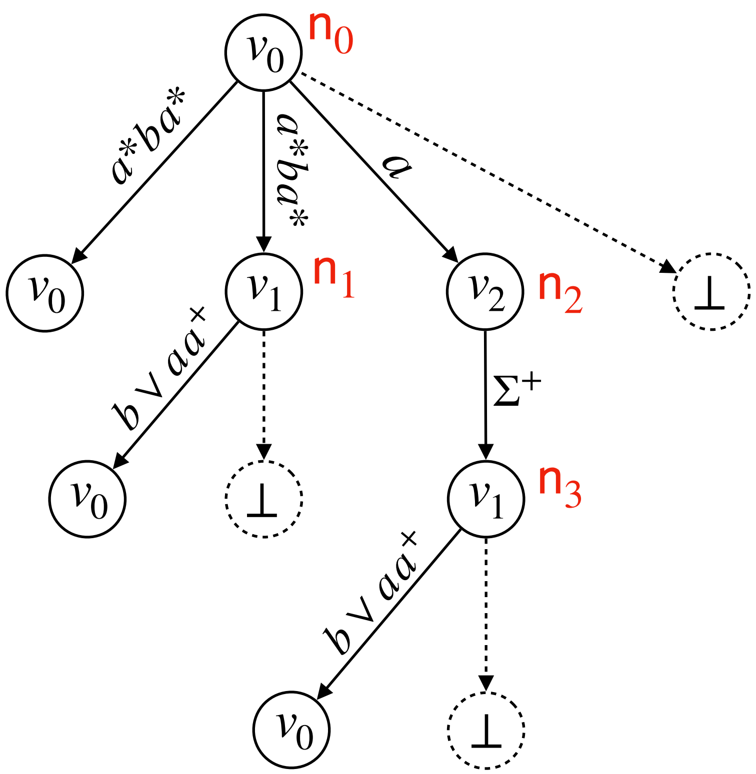
Coalition has a winning strategy in $\mathscr{A}$ ⟺ Coalition has a winning strategy in $\mathscr{T}$

- Proof idea: any history V^{+} in $\mathscr{A}$ uniquely maps to an internal node in $\mathscr{T}$.

Safe coalition problem reduces to existence of a winning coalition strategy in the finite tree unfolding.

Decidability of safe coalition problem

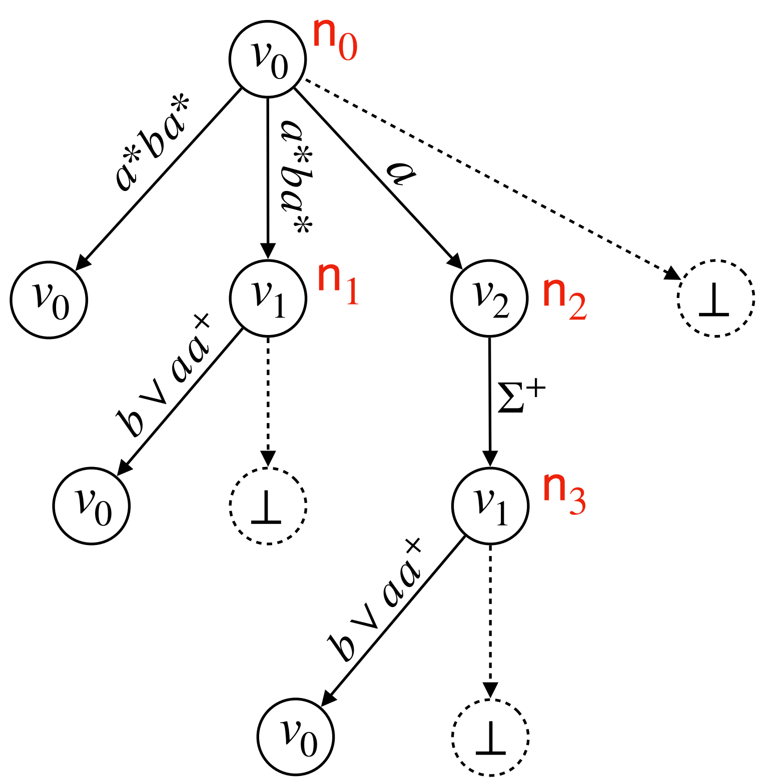
EXPSPACE algorithm



- m = number of internal nodes in $\mathcal{T}$; $m = O(2^{|V|})$.
- r = number of edges in $\mathcal{T}$; $r = O(2^{|V|})$.
- A coalition Strategy in $\mathcal{T}$ is a mapping $\tau : N_{int} \to \Sigma^\omega$.

Decidability of safe coalition problem

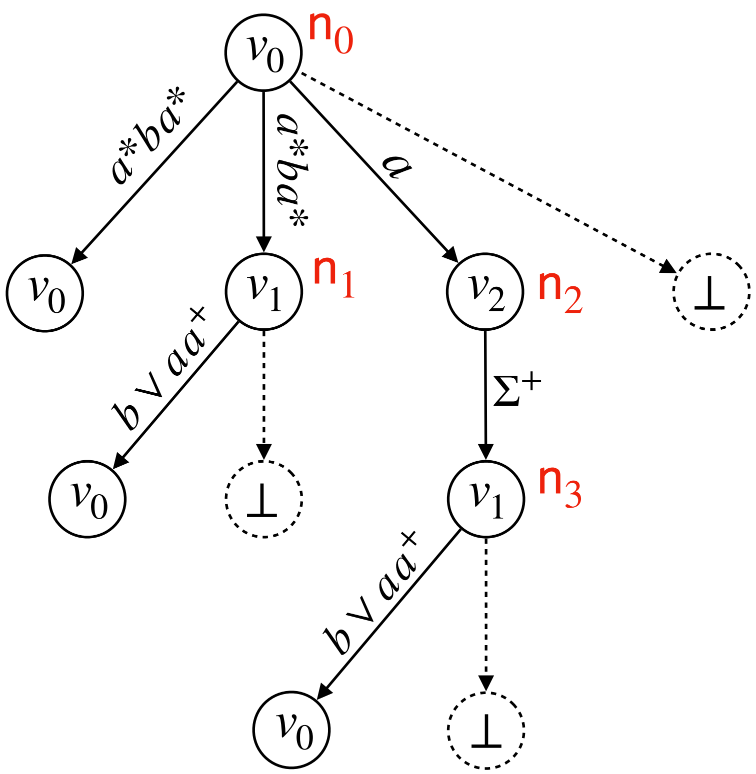
EXPSPACE algorithm



- m = number of internal nodes in $\mathscr{T}$; $m = O(2^{|V|})$.
- r = number of edges in $\mathscr{T}$; $r = O(2^{|V|})$.
- A coalition Strategy in $\mathscr{T}$ is a mapping $\tau : N_{int} \to \Sigma^\omega$.
  - Equivalently, $\tau \in (\Sigma^\omega)^m$.

Decidability of safe coalition problem

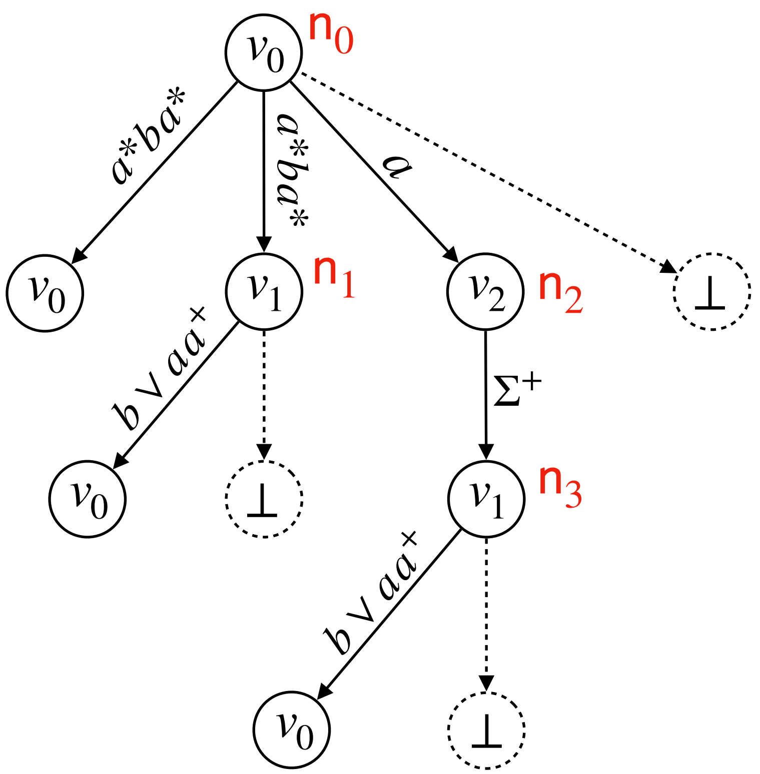
EXPSPACE algorithm



- m = number of internal nodes in $\mathscr{T}$; $m = O(2^{|V|})$.
- r = number of edges in $\mathscr{T}$; $r = O(2^{|V|})$.
- A coalition Strategy in $\mathscr{T}$ is a mapping $\tau : N_{int} \to \Sigma^{\omega}$.
  - Equivalently, $\tau \in (\Sigma^{\omega})^{m}$.
  - Equivalently, $\tau \in (\Sigma^{m})^{\omega}$.

Decidability of safe coalition problem

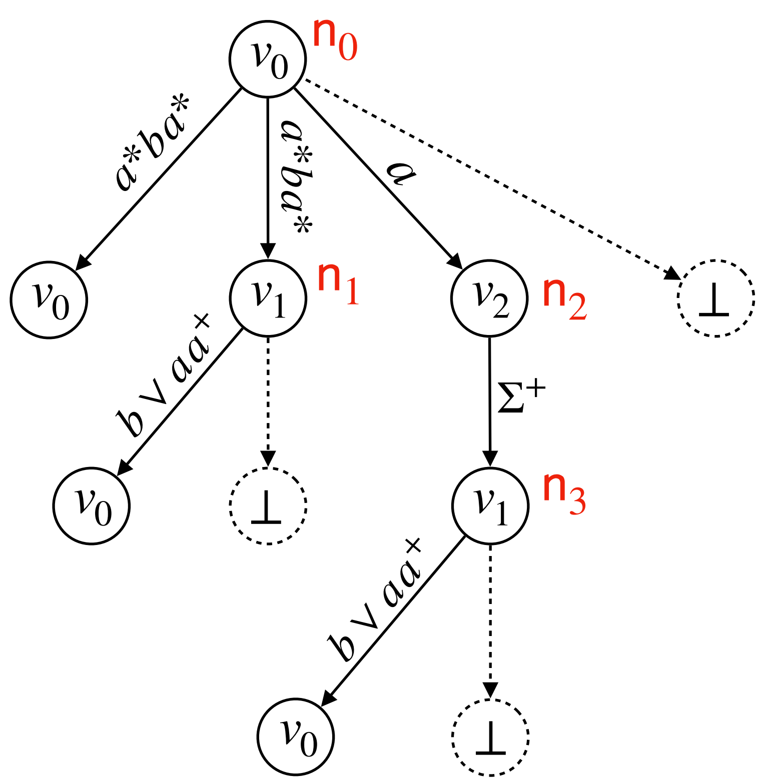
EXPSPACE algorithm



- m = number of internal nodes in $\mathcal{T}$; $m = O(2^{|V|})$.
- r = number of edges in $\mathcal{T}$; $r = O(2^{|V|})$.
- A coalition Strategy in $\mathcal{T}$ is a mapping $\tau : N_{int} \to \Sigma^\omega$.
  - Equivalently, $\tau \in (\Sigma^\omega)^m$.
  - Equivalently, $\tau \in (\Sigma^m)^\omega$.
- Construct safety automaton $\mathcal{B}$ over alphabet Σ^m:
  - runs automata on the edges in parallel.

Decidability of safe coalition problem

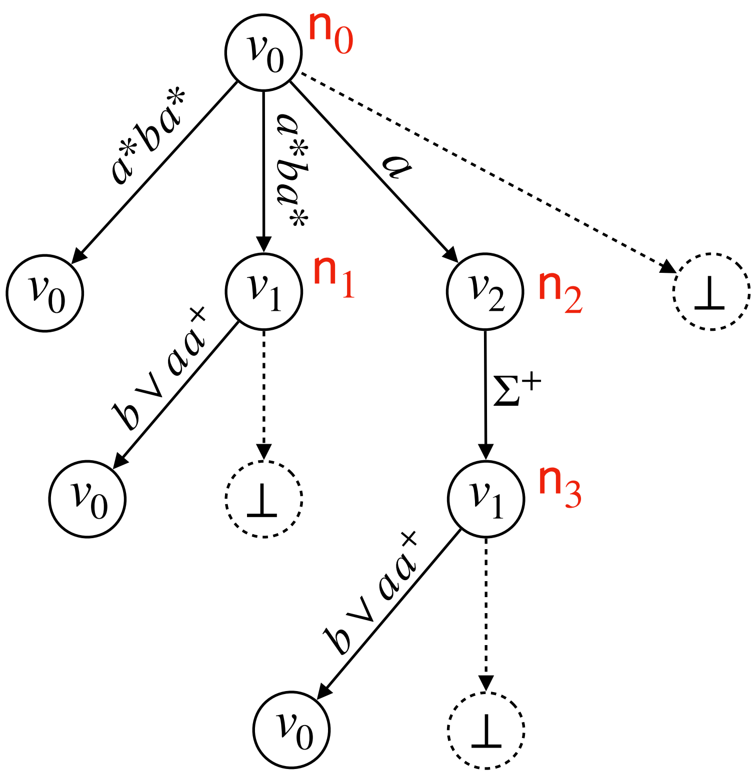
EXPSPACE algorithm



- m = number of internal nodes in $\mathscr{T}$; $m = O(2^{|V|})$.
- r = number of edges in $\mathscr{T}$; $r = O(2^{|V|})$.
- A coalition Strategy in $\mathscr{T}$ is a mapping $\tau : N_{int} \to \Sigma^{\omega}$.
  - Equivalently, $\tau \in (\Sigma^{\omega})^{m}$.
  - Equivalently, $\tau \in (\Sigma^{m})^{\omega}$.
- Construct safety automaton $\mathscr{B}$ over alphabet Σ^{m} :
  - runs automata on the edges in parallel.
  - a (global) state is an r tuple of (local) states.
  - a (global) state corresponds to different branches.

Decidability of safe coalition problem

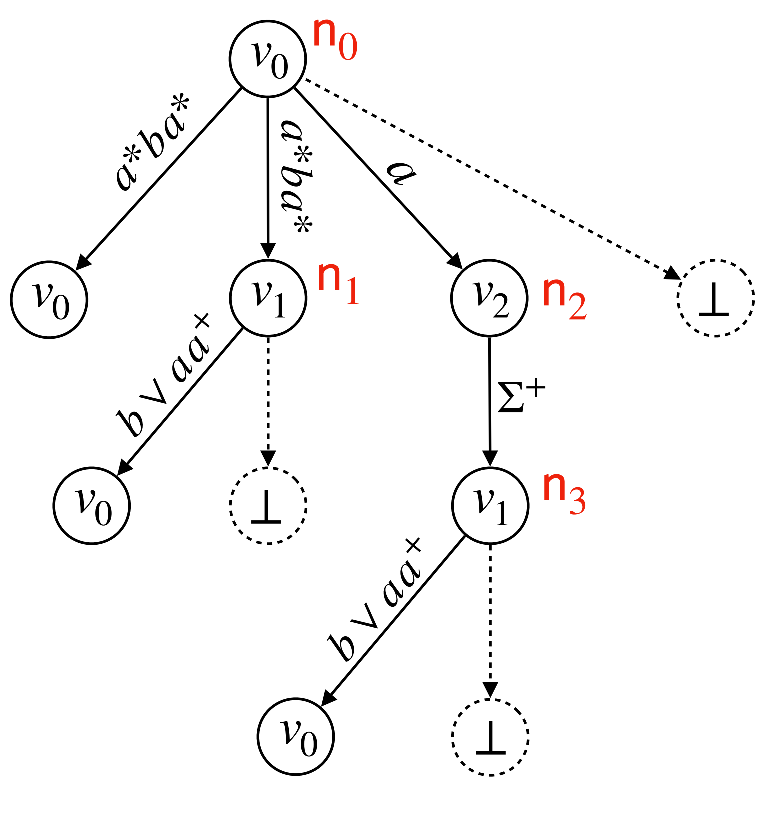
EXPSPACE algorithm



- m = number of internal nodes in $\mathcal{T}$; $m = O(2^{|V|})$.
- r = number of edges in $\mathcal{T}$; $r = O(2^{|V|})$.
- A coalition Strategy in $\mathcal{T}$ is a mapping $\tau : N_{int} \to \Sigma^\omega$.
  - Equivalently, $\tau \in (\Sigma^\omega)^m$.
  - Equivalently, $\tau \in (\Sigma^m)^\omega$.
- Construct safety automaton $\mathcal{B}$ over alphabet Σ^m :
  - runs automata on the edges in parallel.
  - a (global) state is an r tuple of (local) states.
  - a (global) state corresponds to different branches.
  - accepting if the corresponding branches reach safe leaves.

Decidability of safe coalition problem

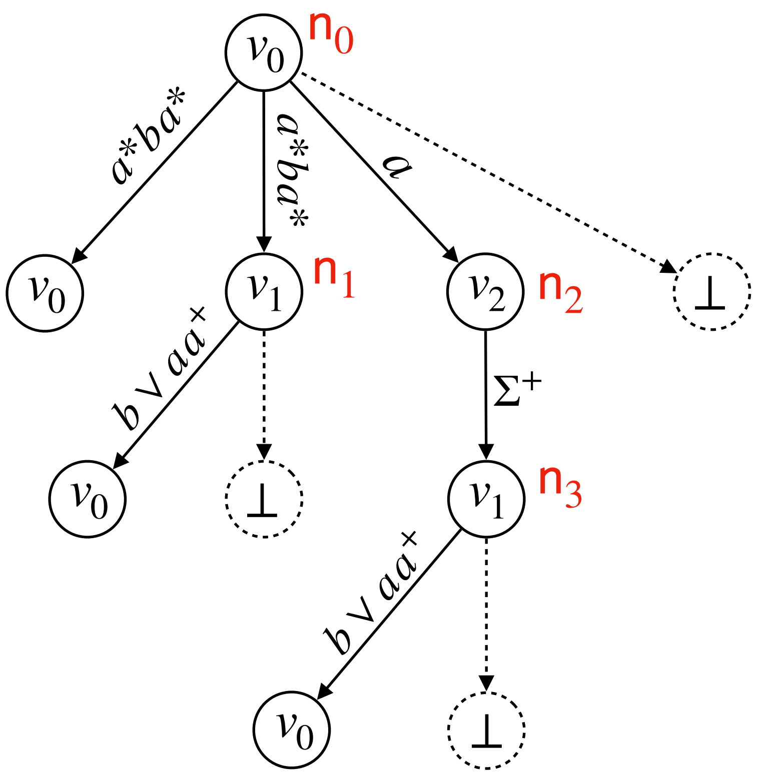
EXPSPACE algorithm



- m = number of internal nodes in $\mathcal{T}$; $m = O(2^{|V|})$.
- r = number of edges in $\mathcal{T}$; $r = O(2^{|V|})$.
- A coalition Strategy in $\mathcal{T}$ is a mapping $\tau : N_{int} \to \Sigma^\omega$.
  - Equivalently, $\tau \in (\Sigma^\omega)^m$.
  - Equivalently, $\tau \in (\Sigma^m)^\omega$.
- Construct safety automaton $\mathscr{B}$ over alphabet Σ^m :
  - runs automata on the edges in parallel.
  - a (global) state is an r tuple of (local) states.
  - a (global) state corresponds to different branches.
  - accepting if the corresponding branches reach safe leaves.
- Accepts words corresponding to winning strategies.

Decidability of safe coalition problem

EXPSPACE algorithm

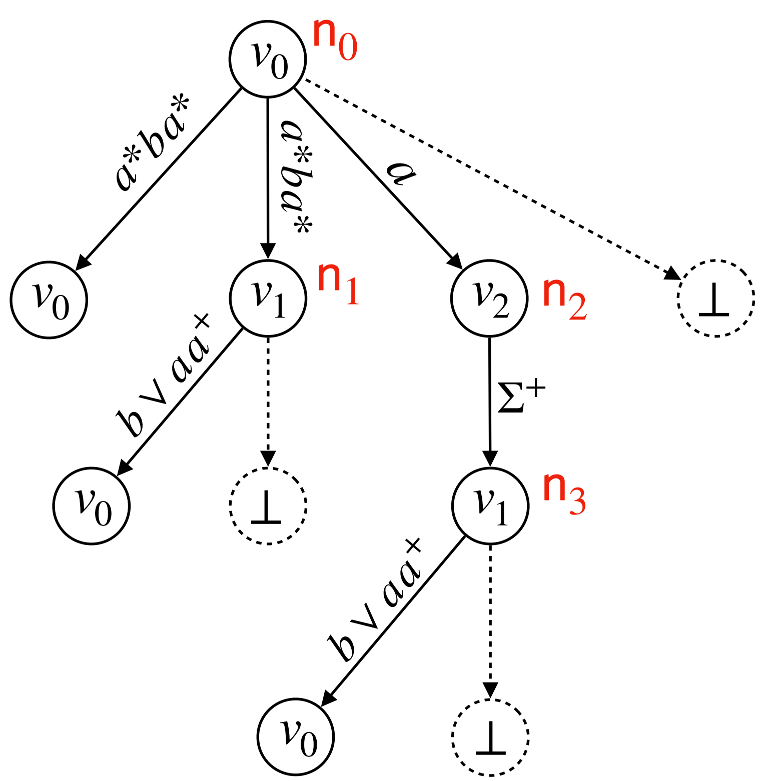


- m = number of internal nodes in $\mathscr{T}$; $m = O(2^{|V|})$.
- r = number of edges in $\mathscr{T}$; $r = O(2^{|V|})$.
- A coalition Strategy in $\mathscr{T}$ is a mapping $\tau : N_{int} \to \Sigma^\omega$.
  - Equivalently, $\tau \in (\Sigma^\omega)^m$.
  - Equivalently, $\tau \in (\Sigma^m)^\omega$.
- Construct safety automaton $\mathscr{B}$ over alphabet Σ^m :
  - runs automata on the edges in parallel.
  - a (global) state is an r tuple of (local) states.
  - a (global) state corresponds to different branches.
  - accepting if the corresponding branches reach safe leaves.
- Accepts words corresponding to winning strategies.

Coalition has a winning strategy in $\mathscr{T}$ ⟺ $\mathscr{L}(\mathscr{B}) \neq \emptyset$

Decidability of safe coalition problem

EXPSPACE algorithm



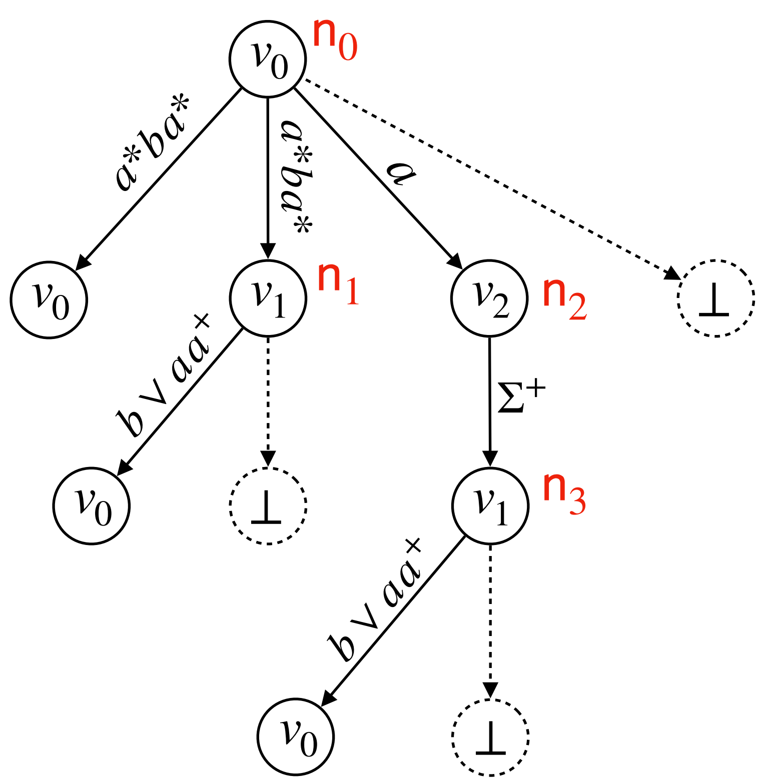
- m = number of internal nodes in $\mathscr{T}$; $m = O(2^{|V|})$.
- r = number of edges in $\mathscr{T}$; $r = O(2^{|V|})$.
- A coalition Strategy in $\mathscr{T}$ is a mapping $\tau : N_{int} \to \Sigma^\omega$.
  - Equivalently, $\tau \in (\Sigma^\omega)^m$.
  - Equivalently, $\tau \in (\Sigma^m)^\omega$.
- Construct safety automaton $\mathscr{B}$ over alphabet Σ^m :
  - runs automata on the edges in parallel.
  - a (global) state is an r tuple of (local) states.
  - a (global) state corresponds to different branches.
  - accepting if the corresponding branches reach safe leaves.
- Accepts words corresponding to winning strategies.

Coalition has a winning strategy in $\mathscr{T}$ ⟺ $\mathscr{L}(\mathscr{B}) \neq \emptyset$

- $|\mathscr{B}| = O(2^{2^{|V|}})$.

Decidability of safe coalition problem

EXPSPACE algorithm



- m = number of internal nodes in $\mathscr{T}$; $m = O(2^{|V|})$.
- r = number of edges in $\mathscr{T}$; $r = O(2^{|V|})$.
- A coalition Strategy in $\mathscr{T}$ is a mapping $\tau : N_{int} \to \Sigma^\omega$.
  - Equivalently, $\tau \in (\Sigma^\omega)^m$.
  - Equivalently, $\tau \in (\Sigma^m)^\omega$.
- Construct safety automaton $\mathscr{B}$ over alphabet Σ^m :
  - runs automata on the edges in parallel.
  - a (global) state is an r tuple of (local) states.
  - a (global) state corresponds to different branches.
  - accepting if the corresponding branches reach safe leaves.
- Accepts words corresponding to winning strategies.

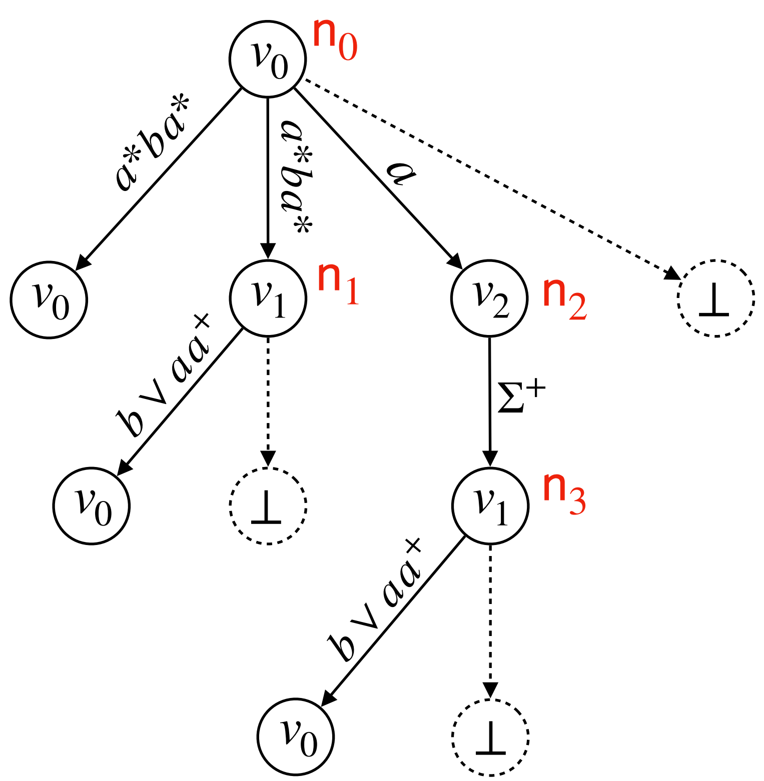
Coalition has a winning strategy in $\mathscr{T}$	⟺	$\mathscr{L}(\mathscr{B}) \neq \varnothing$

- $|\mathscr{B}| = O(2^{2^{|V|}})$.

Safe coalition problem is in **EXPSPACE**

Decidability of safe coalition problem

EXPSPACE algorithm



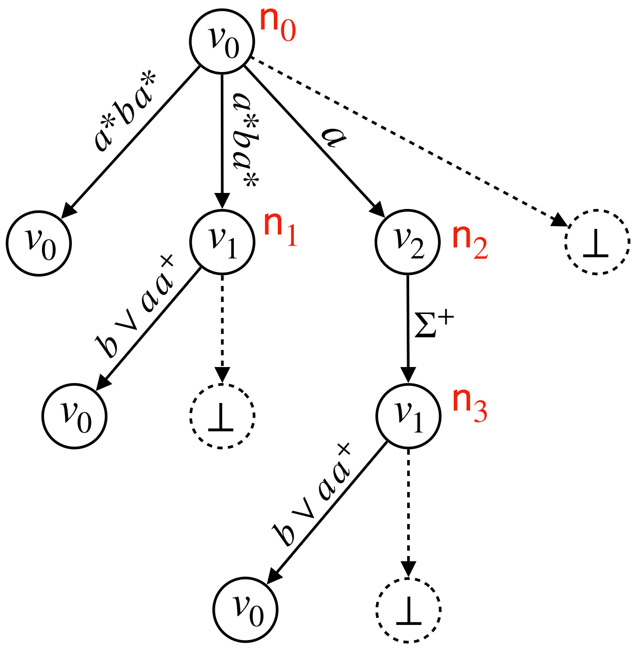
- m = number of internal nodes in $\mathcal{T}$; $m = O(2^{|V|})$.
- r = number of edges in $\mathcal{T}$; $r = O(2^{|V|})$.
- A coalition Strategy in $\mathcal{T}$ is a mapping $\tau : N_{int} \to \Sigma^\omega$.
  - Equivalently, $\tau \in (\Sigma^\omega)^m$.
  - Equivalently, $\tau \in (\Sigma^m)^\omega$.
- Construct safety automaton $\mathcal{B}$ over alphabet Σ^m:
  - runs automata on the edges in parallel.
  - a (global) state is an r tuple of (local) states.
  - a (global) state corresponds to different branches.
  - accepting if the corresponding branches reach safe leaves.
- Accepts words corresponding to winning strategies.

Coalition has a winning strategy in $\mathcal{T}$ ⟺ $\mathcal{L}(\mathcal{B}) \neq \emptyset$

- $|\mathcal{B}| = O(2^{2^{|V|}})$.

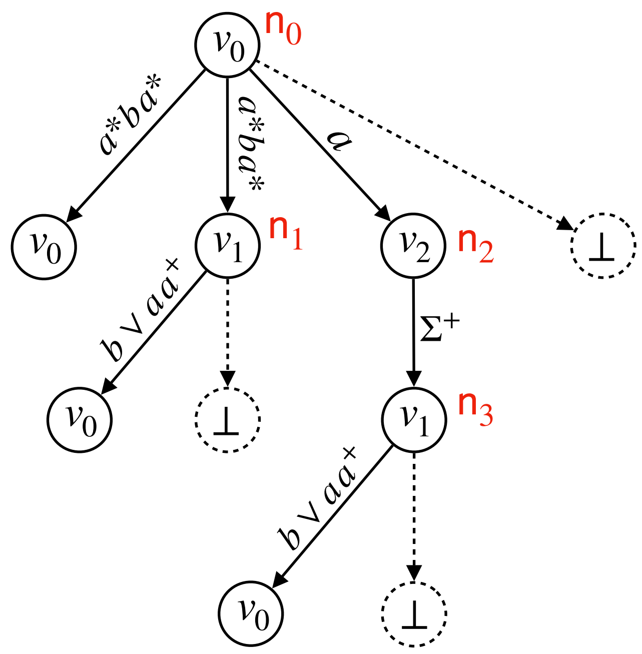
Safe coalition problem is in **EXPSPACE** and PSPACE-hard.

Example

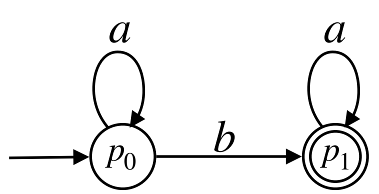


v_0
n_0
a^*ba^*
a^*ba^*
a
v_0
v_1
n_1
v_2
n_2
$\perp$
$b \vee aa^+$
Σ^+
v_0
$\perp$
v_1
n_3
$b \vee aa^+$
v_0
$\perp$

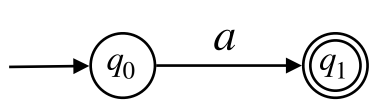
Example



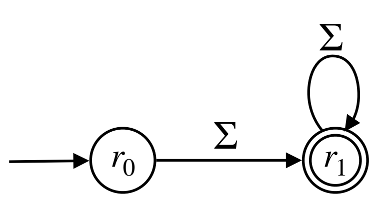
a^*ba^* :



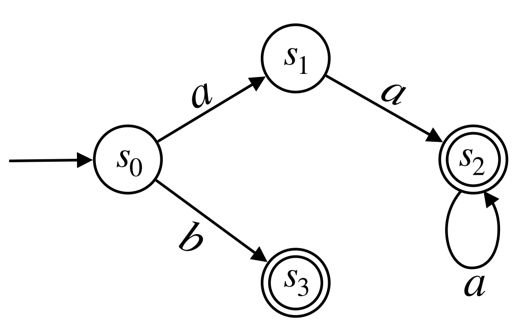
a :



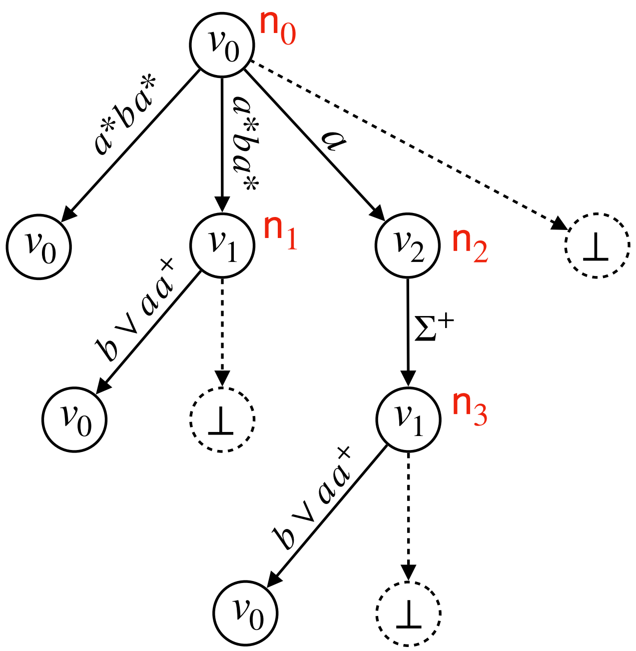
Σ^+ :



$b \vee aa^+$:



Example

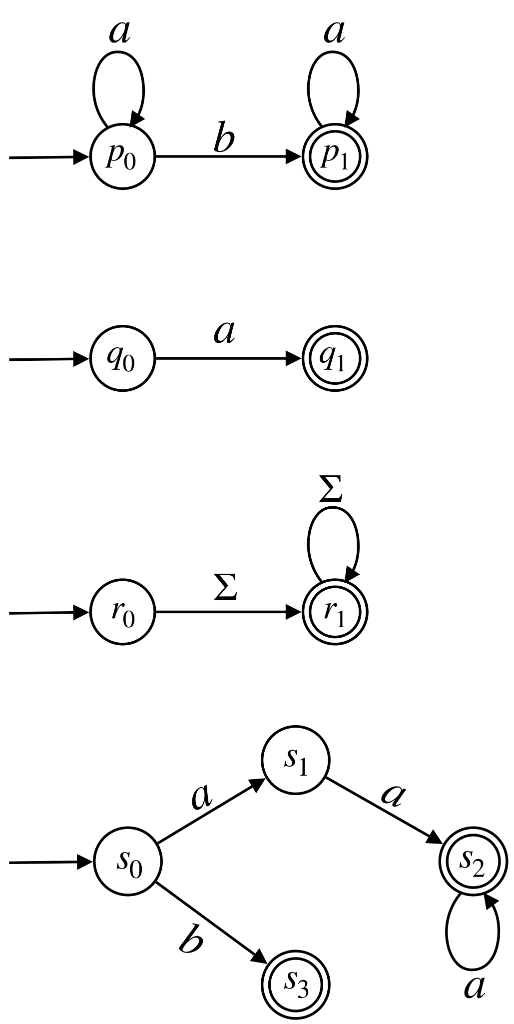


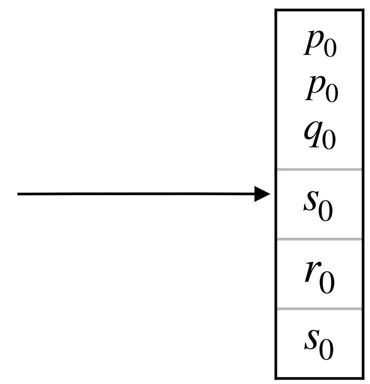
a^*ba^* :

a :

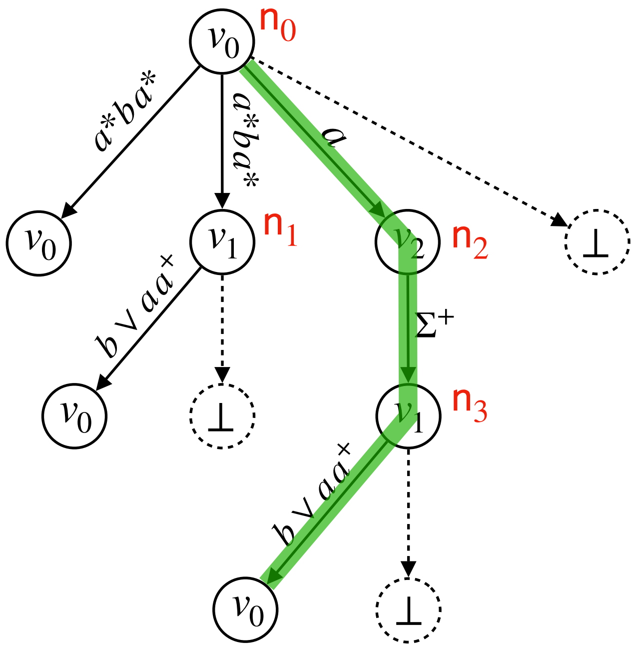
Σ^+ :

$b \vee aa^+$:

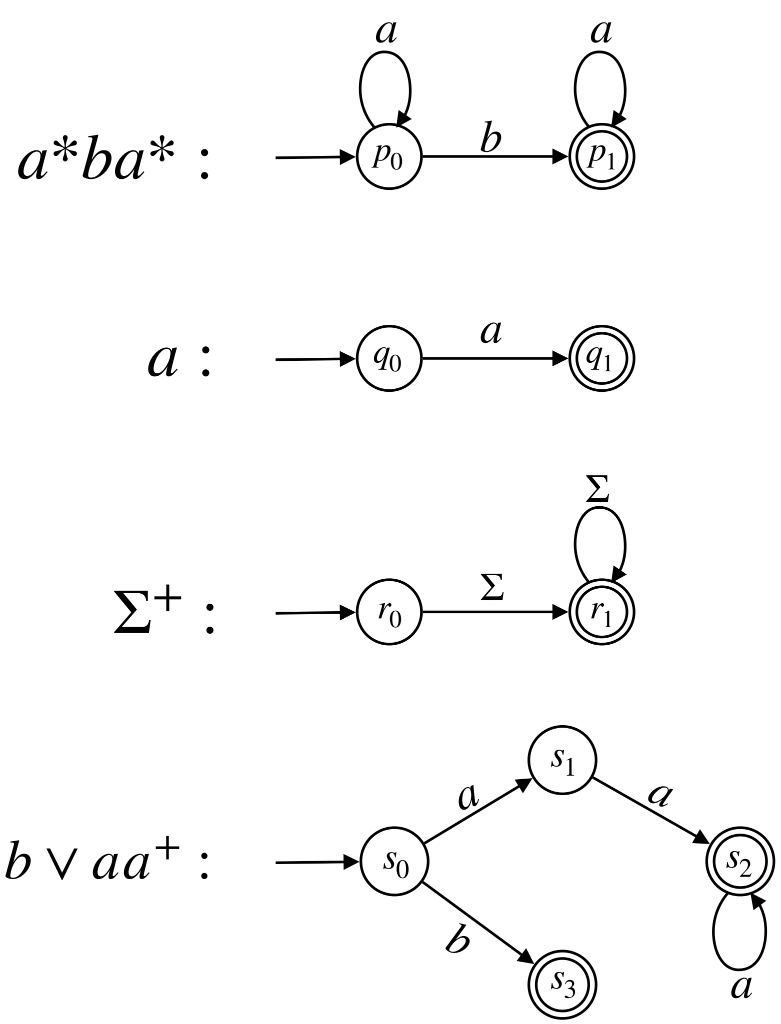




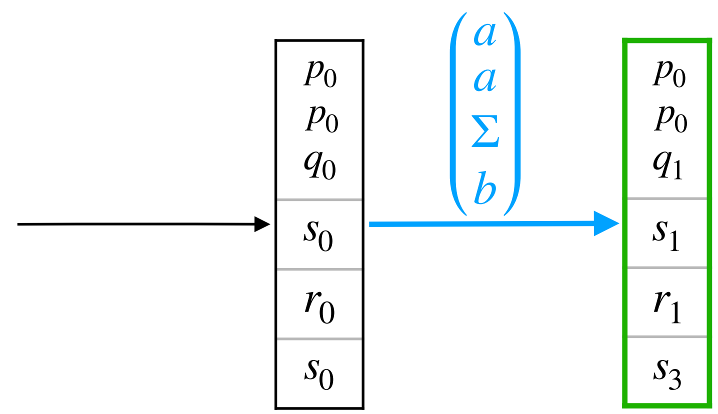
Example



n0
n1
n2
n3
a*ba*
a*ba*
a
Σ+
b ∨ aa+
b ∨ aa+

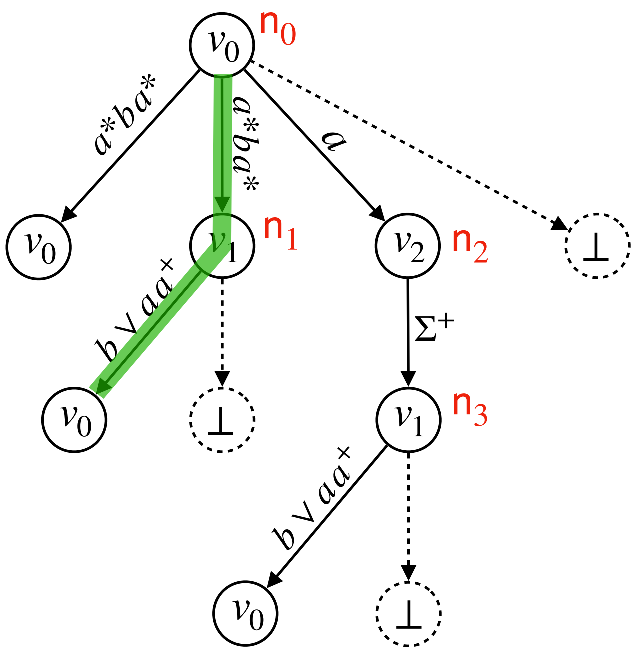


a*ba* :
a
a
b
a :
a
Σ+ :
Σ
Σ
b ∨ aa+ :
a
a
b
a

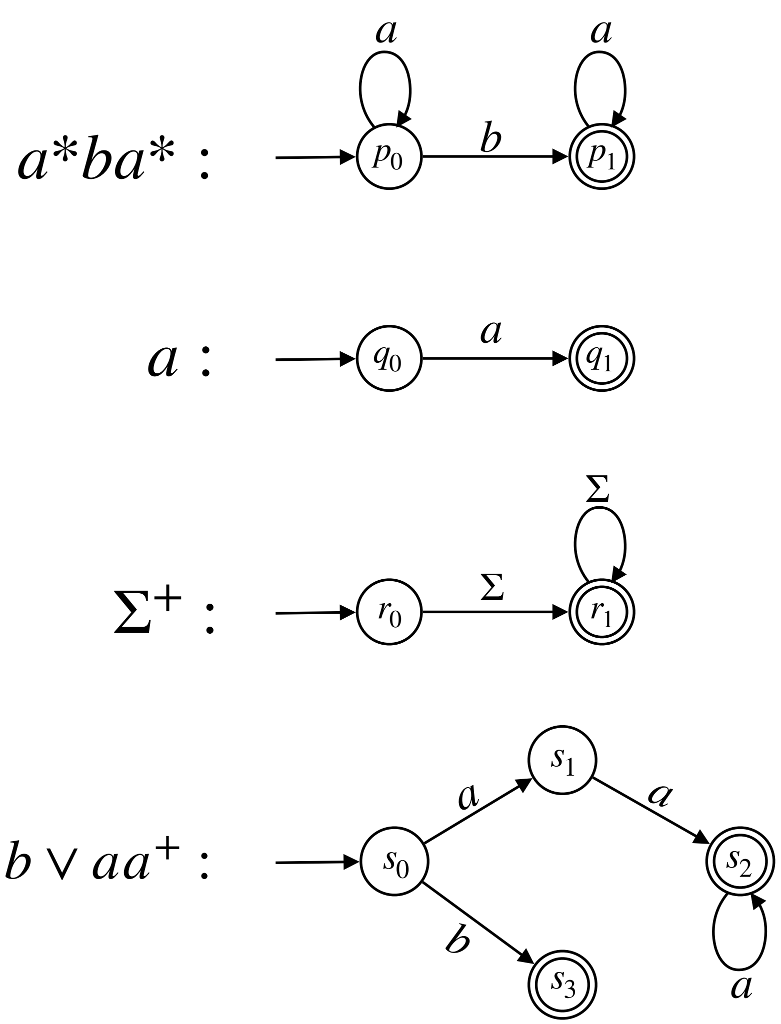


p0
p0
q0
s0
r0
s0
a
a
Σ
b
p0
p0
q1
s1
r1
s3

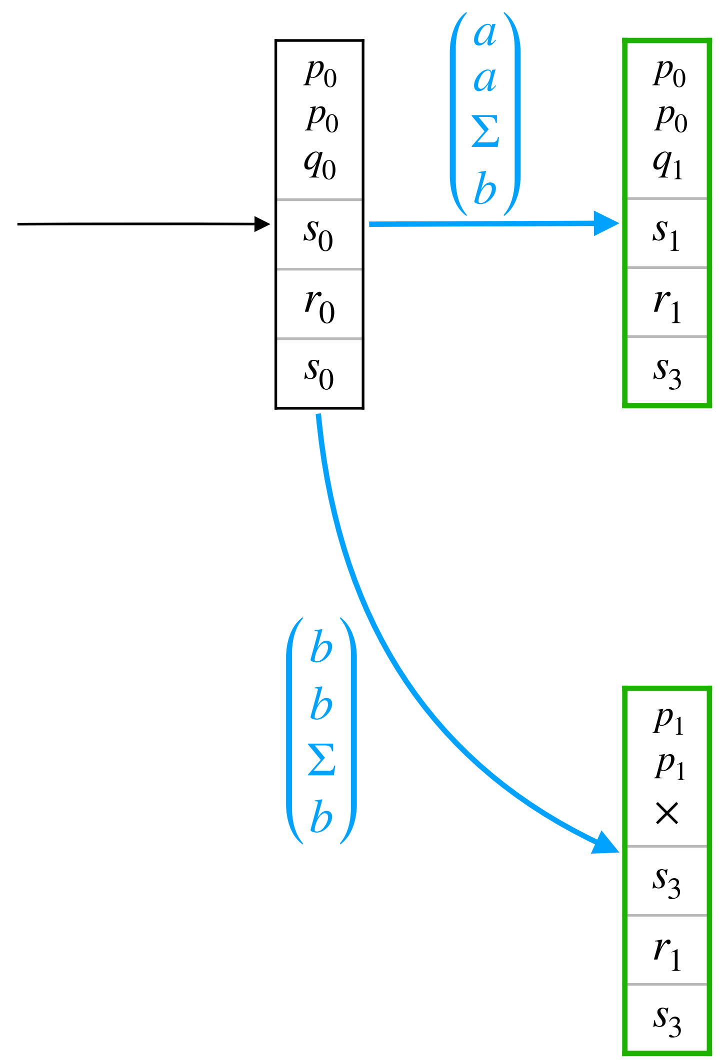
Example



v0
n0
a*ba*
a*ba*
a
v0
v1
n1
v2
n2
⊥
b ∨ aa+
Σ+
v0
⊥
v1
n3
b ∨ aa+
v0
⊥

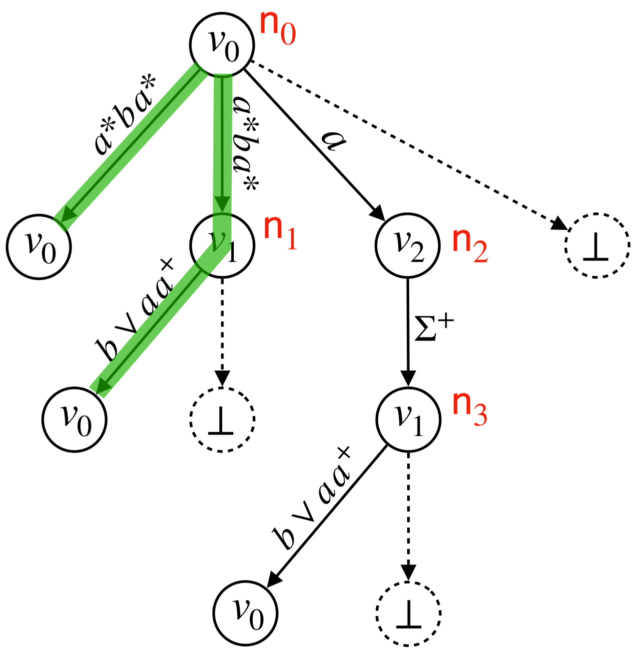


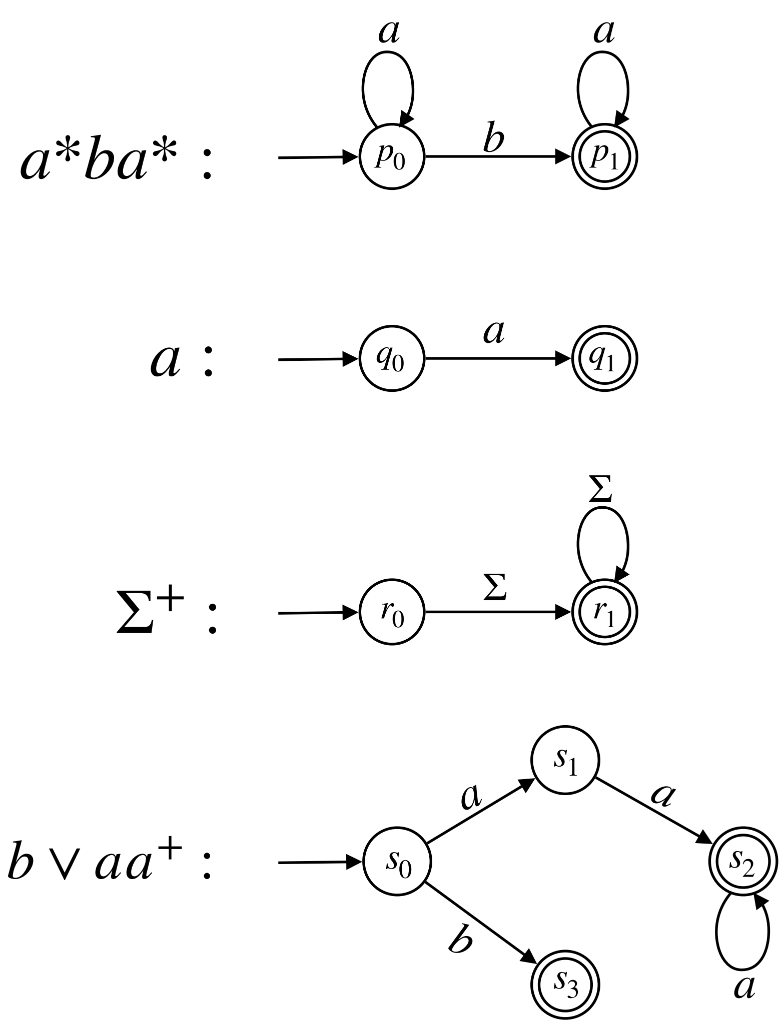
a*ba* :
a
a
b
p0
p1
a :
a
q0
q1
Σ+ :
Σ
Σ
r0
r1
b ∨ aa+ :
a
a
b
a
s0
s1
s2
s3

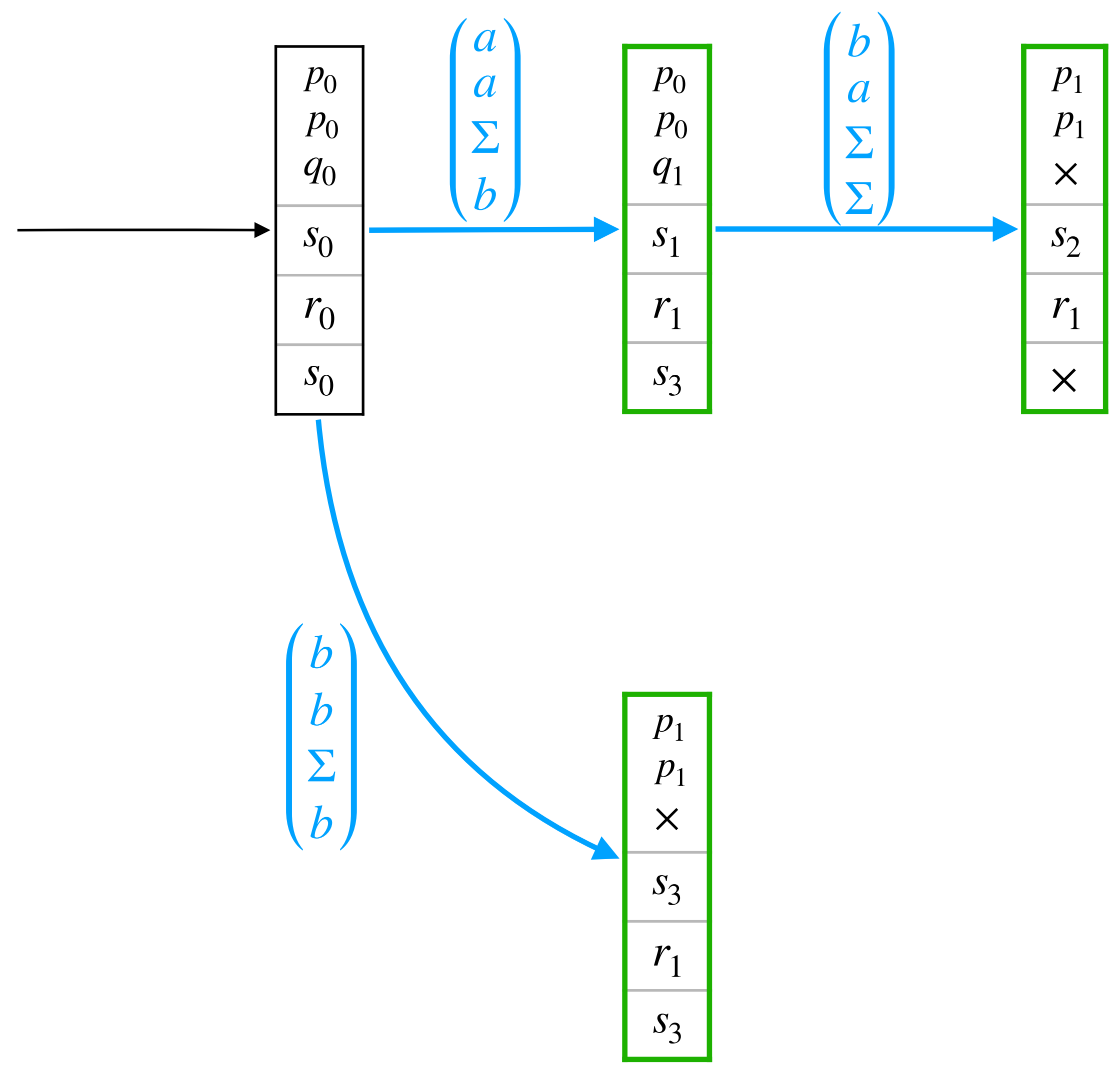


p0
p0
q0
s0
r0
s0
a
a
Σ
b
p0
p0
q1
s1
r1
s3
b
b
Σ
b
p1
p1
×
s3
r1
s3

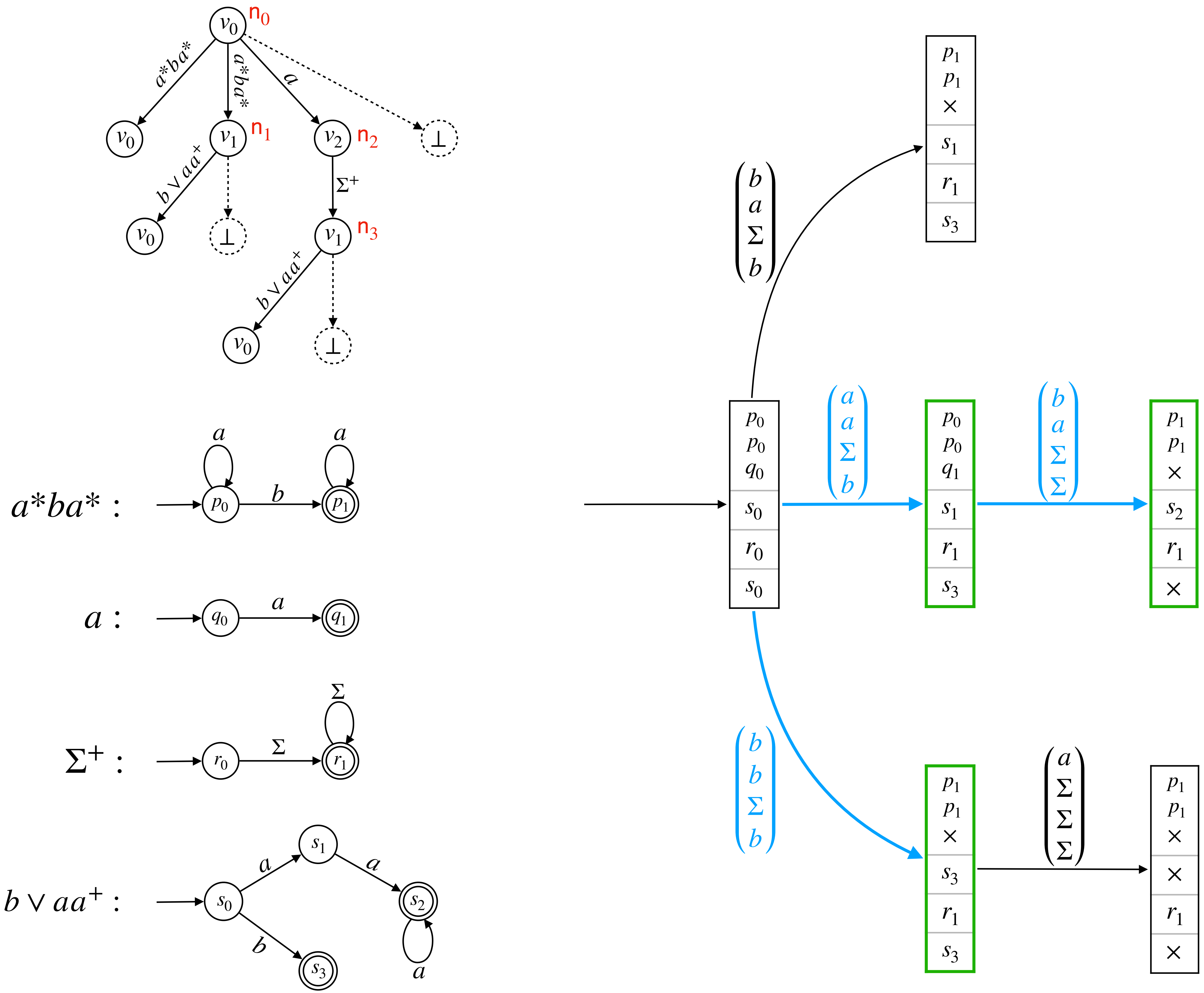
Example



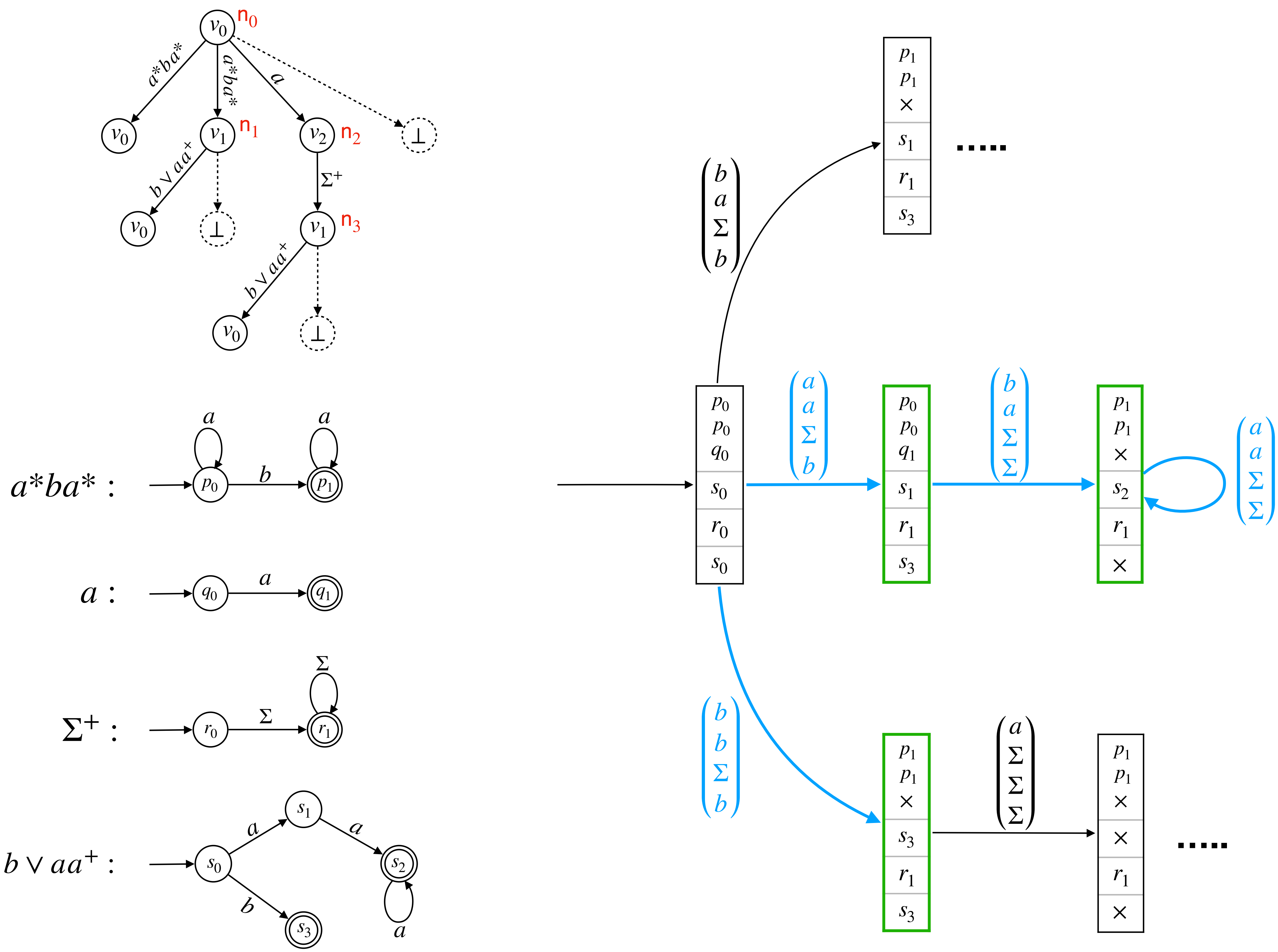




Example

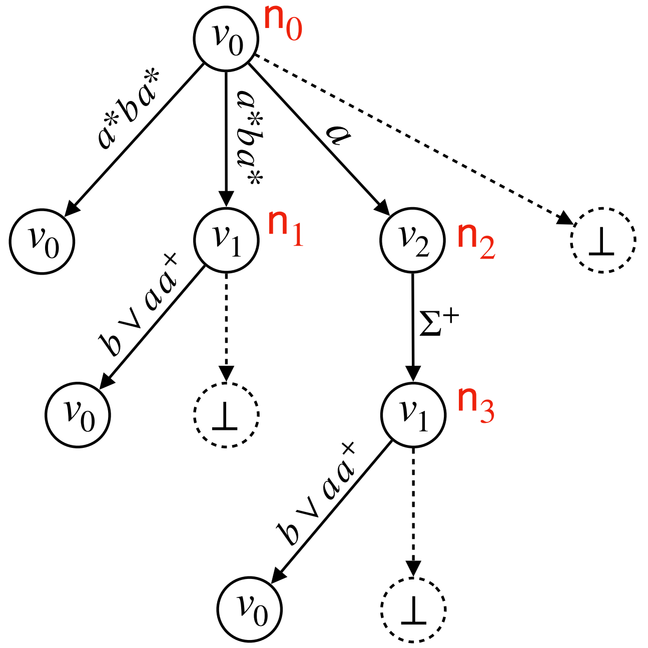


Example



Synthesizing a winning strategy

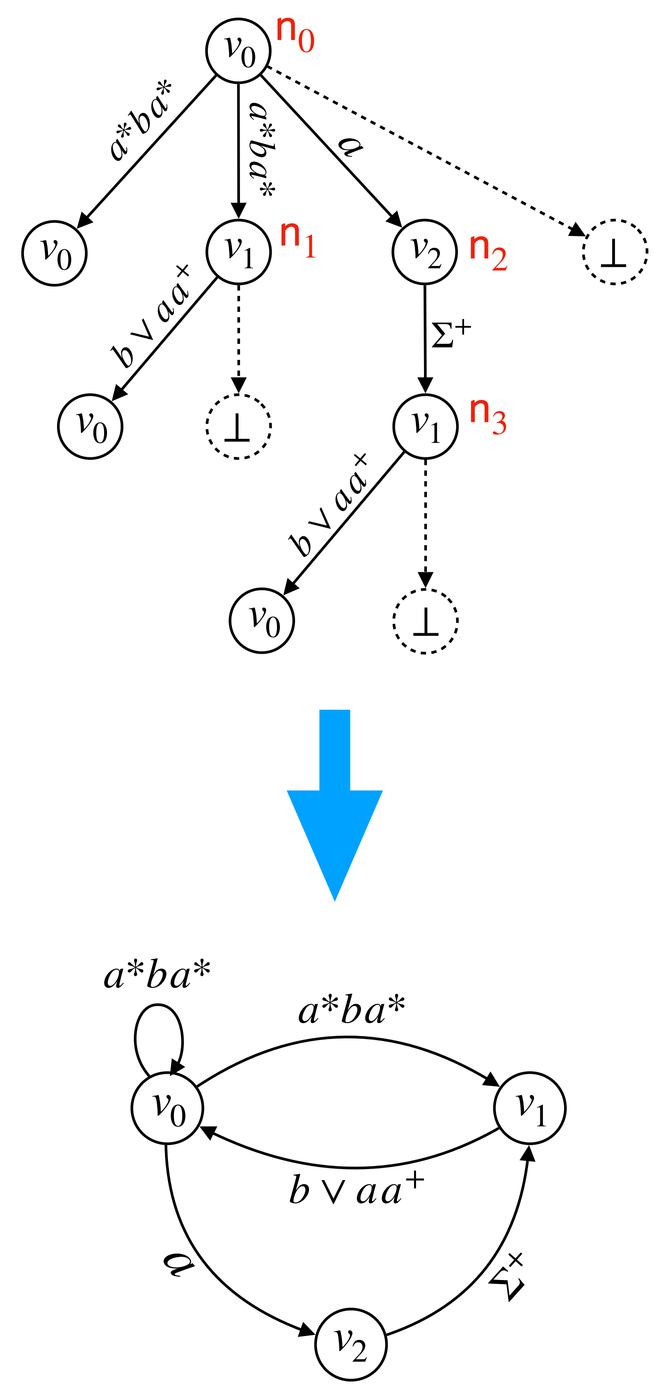
- If $\mathscr{L}(\mathscr{B}) \neq \varnothing$, an accepting word of $\mathscr{B}$ is $\mathsf{u} \,.\, \mathsf{v}^{\omega}$.
- Define: $\lambda(\mathsf{n}_i) = \mathsf{u}_i \,.\, \mathsf{v}_i^{\omega}$
  - λ is a winning strategy in $\mathscr{T}$.



Synthesizing a winning strategy

- If $\mathscr{L}(\mathscr{B}) \neq \emptyset$, an accepting word of $\mathscr{B}$ is $\mathsf{u} \,.\, \mathsf{v}^{\omega}$.
- Define: $\lambda(\mathsf{n}_i) = \mathsf{u}_i \,.\, \mathsf{v}_i^{\omega}$
  - λ is a winning strategy in $\mathscr{T}$.

Transfer λ to a winning strategy $\widetilde{\sigma}$ in $\mathscr{G}$:

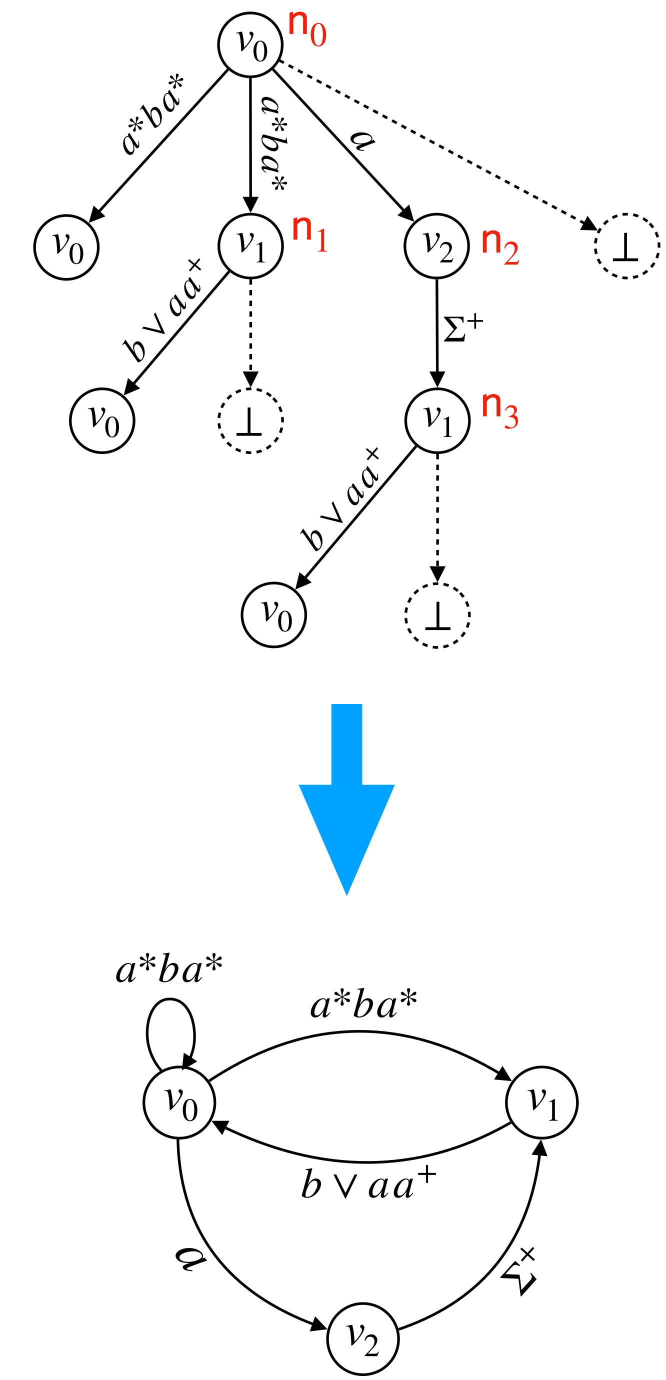


Synthesizing a winning strategy

- If $\mathscr{L}(\mathscr{B}) \neq \emptyset$, an accepting word of $\mathscr{B}$ is $\mathsf{u} \,.\, \mathsf{v}^{\omega}$.
- Define: $\lambda(\mathsf{n}_i) = \mathsf{u}_i \,.\, \mathsf{v}_i^{\omega}$
  - λ is a winning strategy in $\mathscr{T}$.

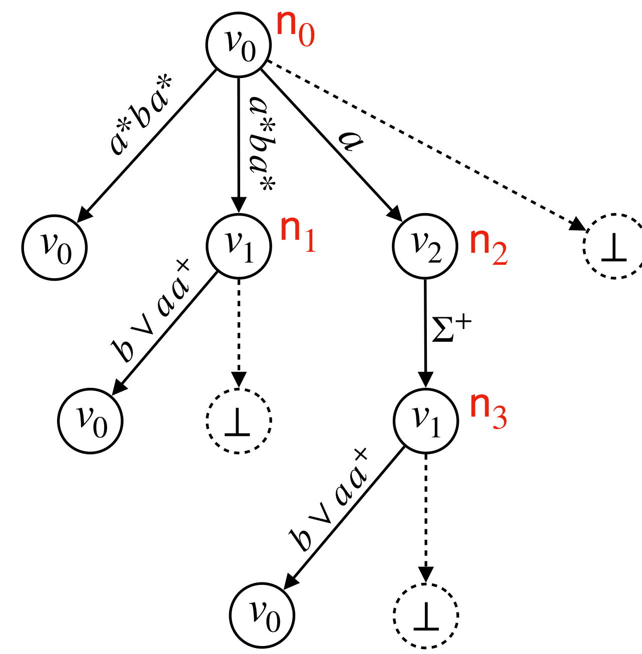
Transfer λ to a **winning** strategy $\widetilde{\sigma}$ in $\mathscr{G}$:

- A history V^+ in $\mathscr{A}$ uniquely maps to an internal node in $\mathscr{T}$ by $h \mapsto \mathsf{zip}(h)$.
- Define $\widetilde{\sigma}$: $\widetilde{\sigma}(h) = \lambda(\mathsf{zip}(h))$
  - $\widetilde{\sigma}$ is a winning strategy in $\mathscr{G}$.

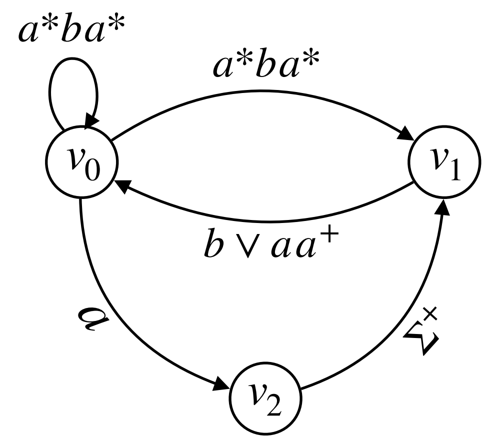


Synthesizing a winning strategy

- If $\mathscr{L}(\mathscr{B}) \neq \varnothing$, an accepting word of $\mathscr{B}$ is $\mathsf{u}\,.\,\mathsf{v}^{\omega}$.
- Define: $\lambda(\mathsf{n}_i) = \mathsf{u}_i\,.\,\mathsf{v}_i^{\omega}$
  - λ is a winning strategy in $\mathscr{T}$.



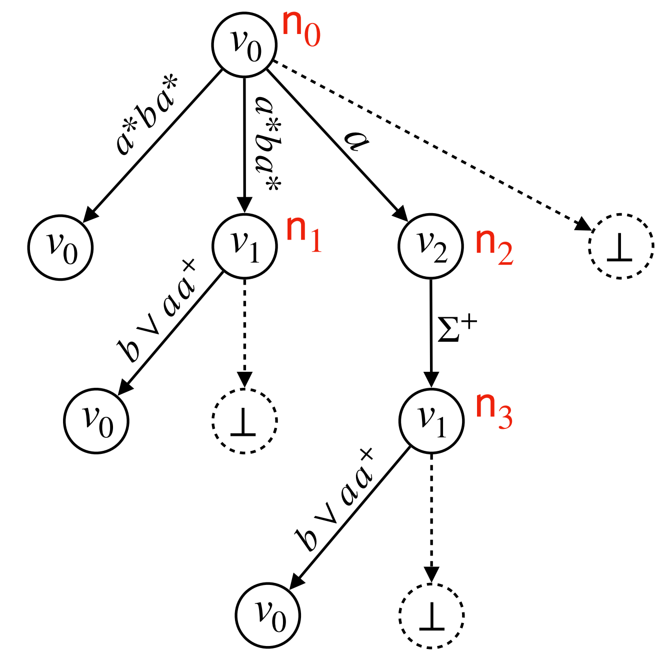
Transfer λ to a winning strategy $\widetilde{\sigma}$ in $\mathscr{G}$:



- A history V^+ in $\mathscr{A}$ uniquely maps to an internal node in $\mathscr{T}$ by $h \mapsto \mathsf{zip}(h)$.
- Define $\widetilde{\sigma}$: $\widetilde{\sigma}(h) = \lambda(\mathsf{zip}(h))$
  - $\widetilde{\sigma}$ is a winning strategy in $\mathscr{G}$.

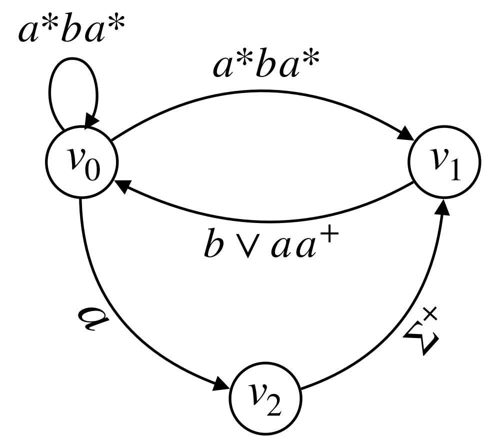
Synthesizing a winning coalition strategy is in **EXPSPACE**.

Synthesizing a winning strategy



- If $\mathscr{L}(\mathscr{B}) \neq \emptyset$, an accepting word of $\mathscr{B}$ is $\mathsf{u} \,.\, \mathsf{v}^\omega$.
- Define: $\lambda(\mathsf{n}_i) = \mathsf{u}_i \,.\, \mathsf{v}_i^\omega$
  - λ is a winning strategy in $\mathscr{T}$.

Transfer λ to a **winning** strategy $\widetilde{\sigma}$ in $\mathscr{G}$:

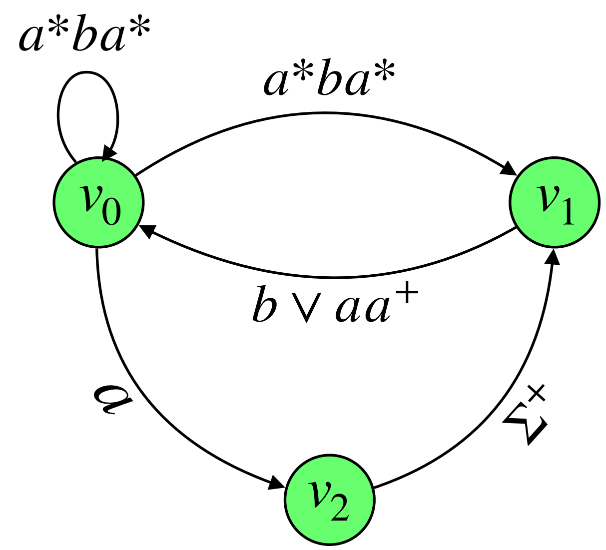


- A history V^+ in $\mathscr{A}$ uniquely maps to an internal node in $\mathscr{T}$ by $h \mapsto \mathsf{zip}(h)$.
- Define $\widetilde{\sigma}$: $\widetilde{\sigma}(h) = \lambda(\mathsf{zip}(h))$
  - $\widetilde{\sigma}$ is a winning strategy in $\mathscr{G}$.

Synthesizing a winning coalition strategy is in **EXPSPACE**.

$\widetilde{\sigma}$ uses memory of size $2^{O(|V|)}$, which is unavoidable.

Conclusion



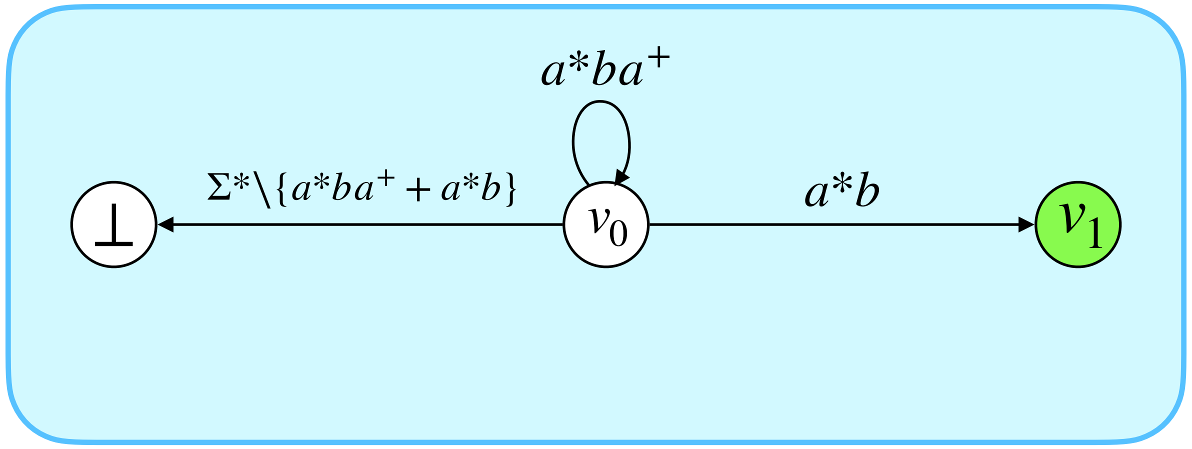
- Parameterized concurrent arena.
- Coalition problem with safety objective.
- Safe coalition problem is decidable in exponential space.
  - Reduce to coalition problem on finite tree unfolding.
  - Construct doubly-exponential size safety automaton.
  - Check non-emptiness.
- Safe coalition problem is PSPACE-hard.
- Synthesizing a winning strategy (if exists) needs exponential space.
- A winning strategy (if exists) needs exponential size memory.

Future work

- Tight complexity bound.

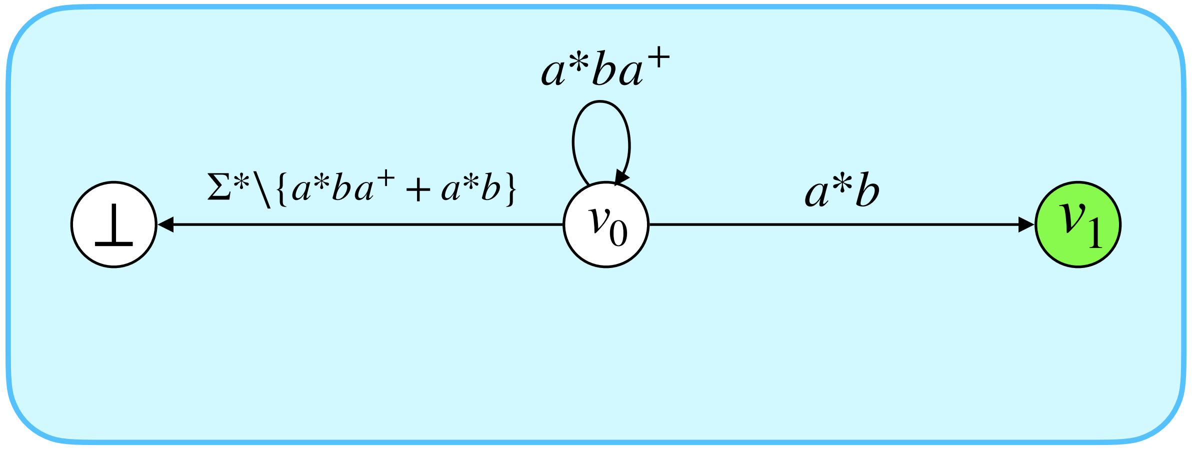
Future work

- Tight complexity bound.
- Reachability condition:



Future work

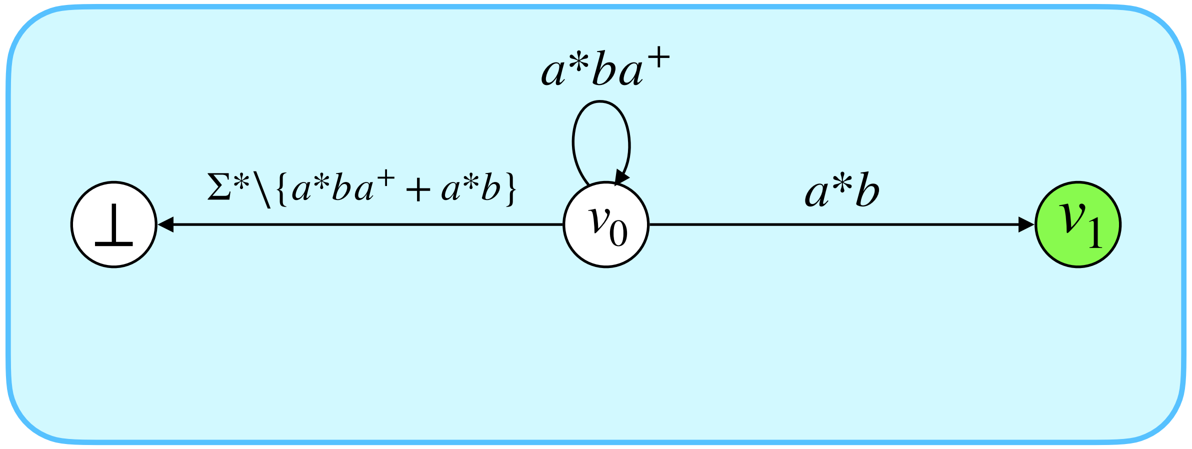
- Tight complexity bound.
- Reachability condition:



- Players collectively want to reach v_1 .
- Number of players unknown.
- Collective winning strategy:
    - Player i plays b at i-th round, a otherwise.

Future work

- Tight complexity bound.
- Reachability condition:



- Players collectively want to reach v_1 .
- Number of players unknown.
- Collective winning strategy:
  - Player i plays b at i-th round, a otherwise.

Thank You