Laboratoire Méthodes Formelles

université PARIS-SACLAY

cnrs

école normale supérieure paris—saclay

Parameterized concurrent games

Patricia Bouyer-Decitre

Laboratoire Méthodes Formelles

université PARIS-SACLAY

cnrs

école normale supérieure paris—saclay

Parameterized concurrent games

Patricia Bouyer-Decitre

Based on joint works with Nathalie Bertrand and Anirban Majumdar

Parameterized concurrent games

Patricia Bouyer-Decitre

Based on joint works with Nathalie Bertrand and Anirban Majumdar

Preliminary results published at FSTTCS'19'20


ETAPS
2018

Goal of this work

Goal of this work

General objective:

How game-theoretic models and technics/tools can help handling parameterized verification and synthesis?

Goal of this work

General objective:

How game-theoretic models and technics/tools can help handling parameterized verification and synthesis?

- Propose a new game-based model for parameterized reasoning

Goal of this work

General objective:

How game-theoretic models and technics/tools can help handling parameterized verification and synthesis?

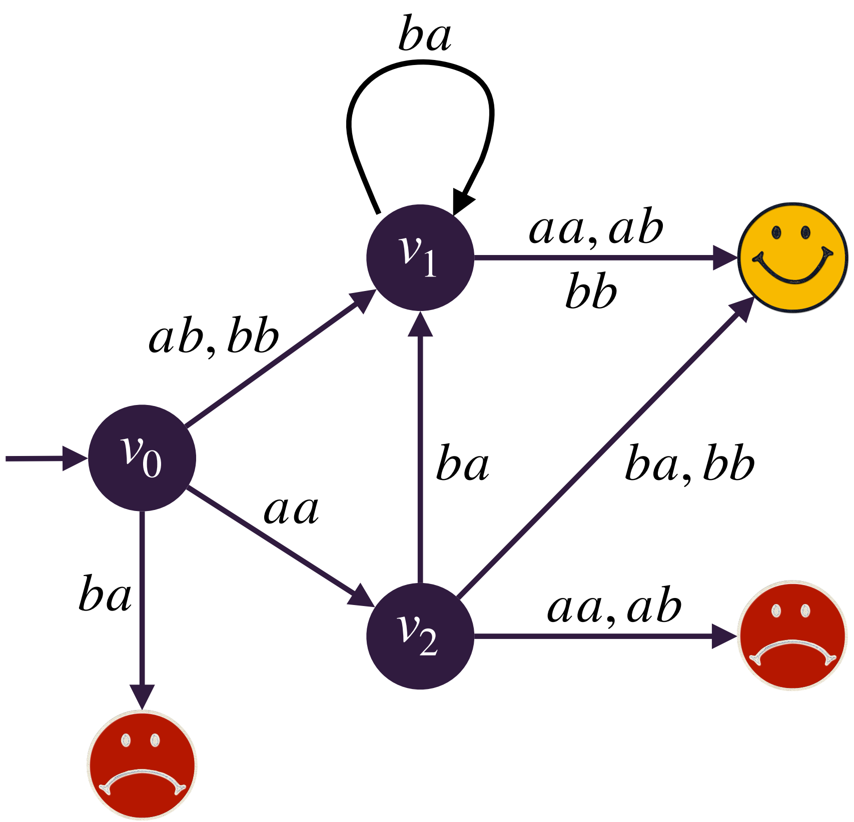
- Propose a new game-based model for parameterized reasoning
- Design synthesis algorithms in two settings:
  - Crowd controller problem
  - Coalition problem

Two-player games as a model for controller synthesis

- Two-player game = model for open systems
- Two players = system *vs* environment
- Winning objective for system player = specification
- Winning strategy for system player = safe controller

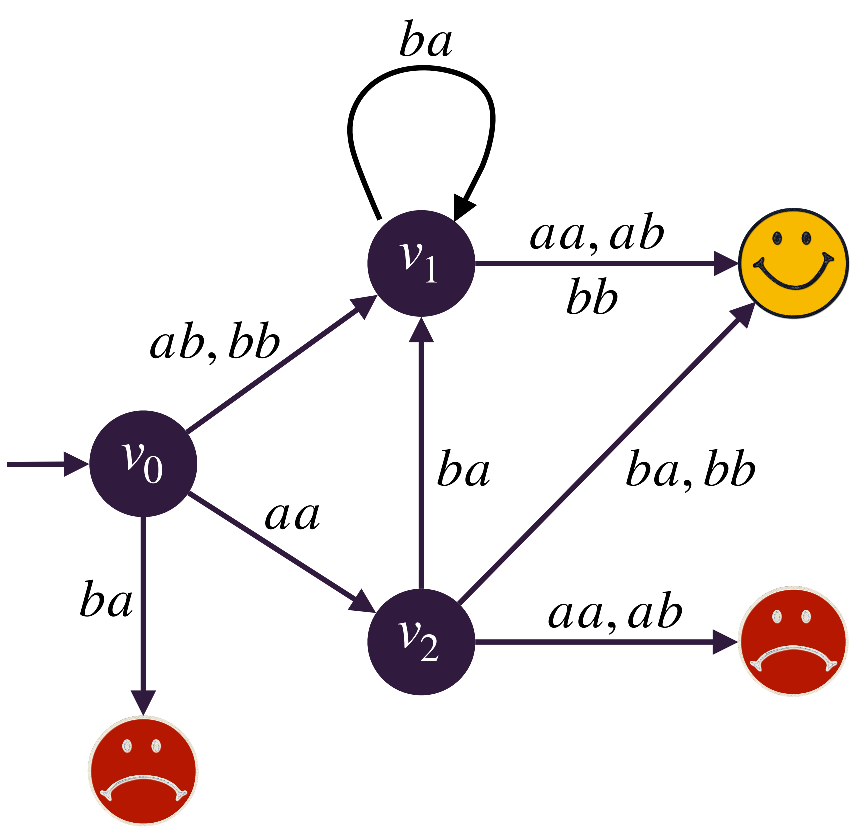
[Tho02] W. Thomas : Infinite Games and Verification (CAV'02)
[FOX] Lectures in Game Theory for Computer Scientists (edited by K. R. Apt and E. Grädel)

Two-player (zero-sum) concurrent games



[AHK98,07] L. De Alfaro, T. Henzinger, O. Kupferman. Concurrent reachability games (FOCS’98, TCS’07)

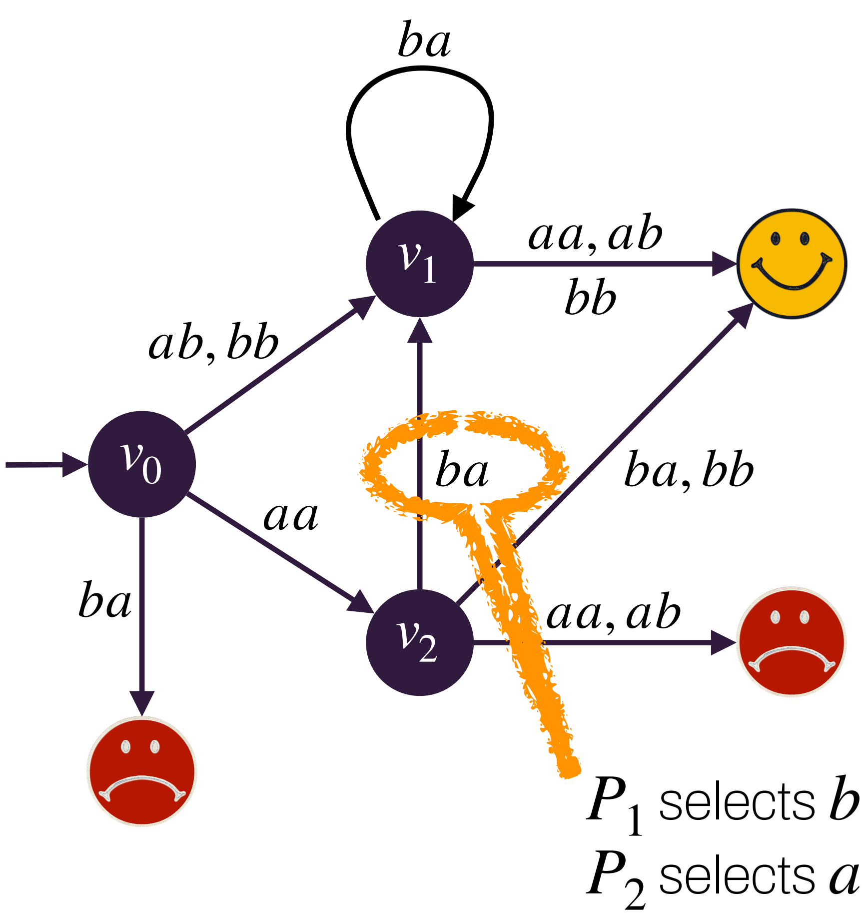
Two-player (zero-sum) concurrent games



- Game graph $G = (V, \delta)$ with $\delta \subseteq V \times \Sigma^2 \times V$

[AHK98,07] L. De Alfaro, T. Henzinger, O. Kupferman. Concurrent reachability games (FOCS'98, TCS'07)

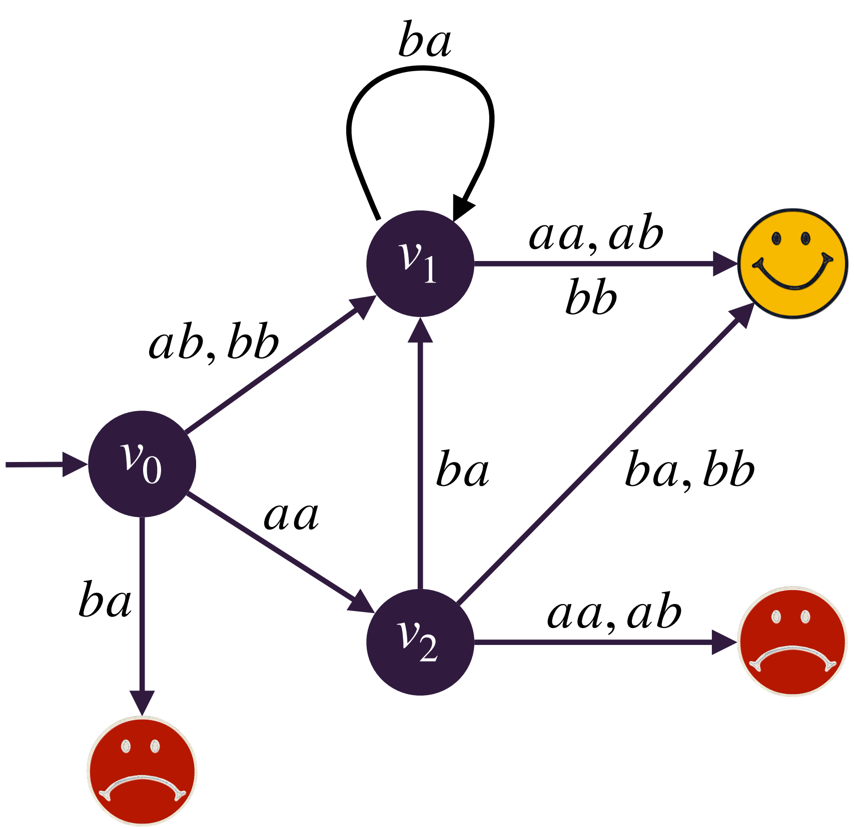
Two-player (zero-sum) concurrent games



- Game graph $G = (V, \delta)$ with $\delta \subseteq V \times \Sigma^2 \times V$

[AHK98,07] L. De Alfaro, T. Henzinger, O. Kupferman. Concurrent reachability games (FOCS'98, TCS'07)

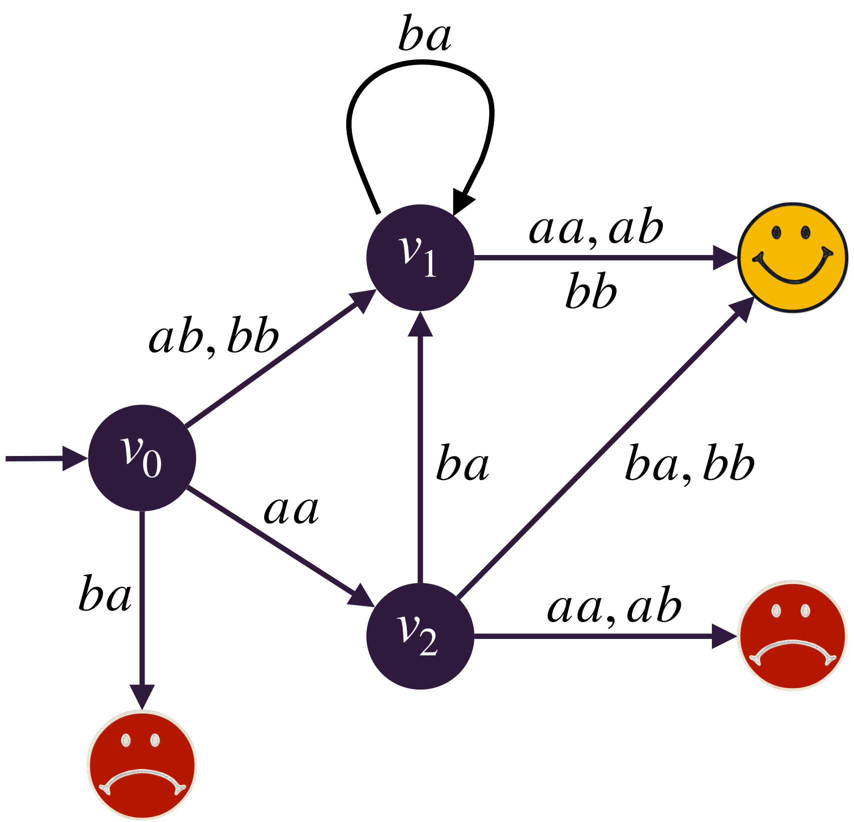
Two-player (zero-sum) concurrent games



- Game graph $G = (V, \delta)$ with $\delta \subseteq V \times \Sigma^2 \times V$
- A strategy for player P_i is a mapping $\sigma_i : V^+ \to \Sigma$

[AHK98,07] L. De Alfaro, T. Henzinger, O. Kupferman. Concurrent reachability games (FOCS'98, TCS'07)

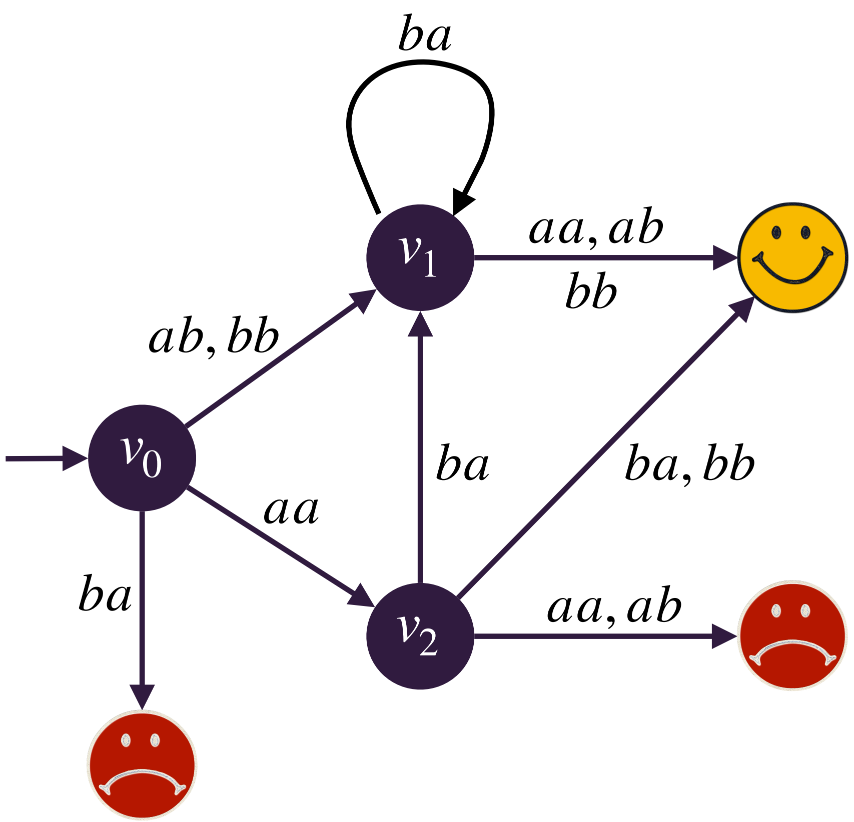
Two-player (zero-sum) concurrent games



- Game graph $G = (V, \delta)$ with $\delta \subseteq V \times \Sigma^2 \times V$
- A strategy for player P_i is a mapping $\sigma_i : V^+ \to \Sigma$
- Given (σ_1, σ_2), there are possibly several outcomes in V^ω. It is unique whenever the arena is deterministic. We write $\mathsf{Out}(\sigma_1, \sigma_2)$ for that set and $\mathsf{Out}(\sigma_i)$ if only σ_i is fixed

[AHK98,07] L. De Alfaro, T. Henzinger, O. Kupferman. Concurrent reachability games (FOCS'98, TCS'07)

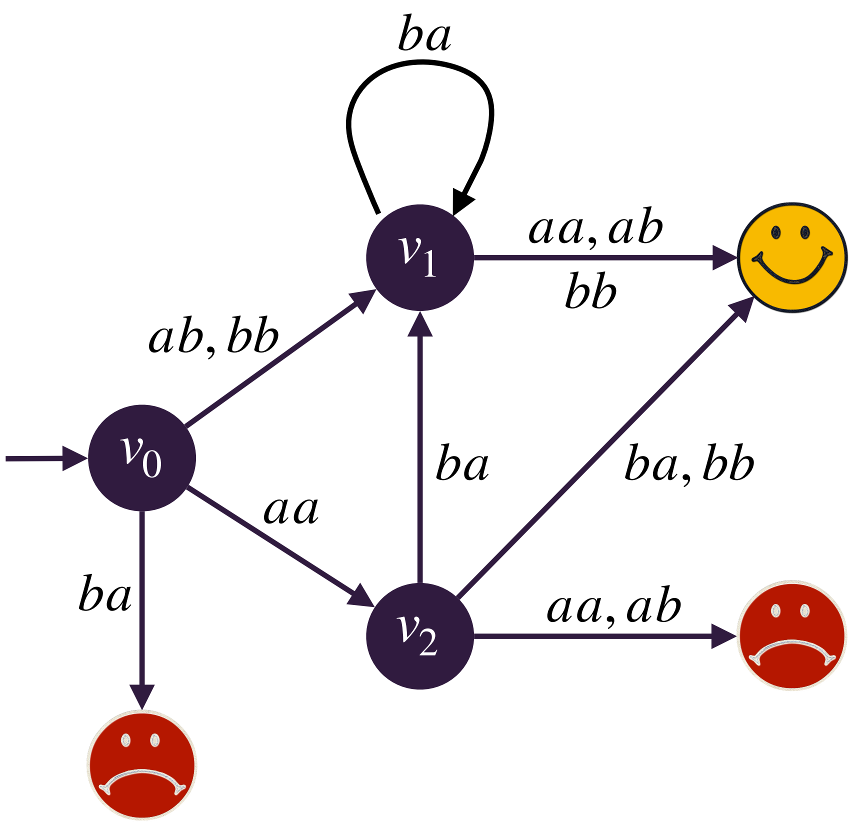
Two-player (zero-sum) concurrent games



- Game graph $G = (V, \delta)$ with $\delta \subseteq V \times \Sigma^2 \times V$
- A strategy for player P_i is a mapping $\sigma_i : V^+ \to \Sigma$
- Given (σ_1, σ_2), there are possibly several outcomes in V^ω. It is unique whenever the arena is deterministic. We write $\mathsf{Out}(\sigma_1, \sigma_2)$ for that set and $\mathsf{Out}(\sigma_i)$ if only σ_i is fixed
- Winning condition for P_1: reach 
- Winning condition for P_2: avoid 

[AHK98,07] L. De Alfaro, T. Henzinger, O. Kupferman. Concurrent reachability games (FOCS'98, TCS'07)

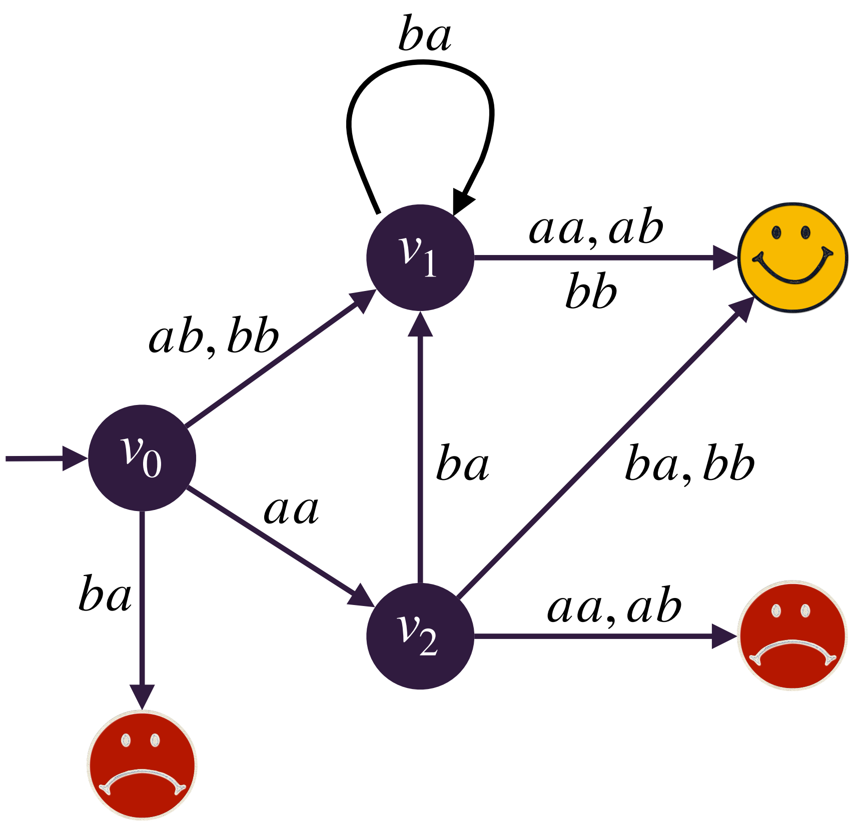
Two-player (zero-sum) concurrent games



- Game graph $G = (V, \delta)$ with $\delta \subseteq V \times \Sigma^2 \times V$
- A strategy for player P_i is a mapping $\sigma_i : V^+ \to \Sigma$
- Given (σ_1, σ_2), there are possibly several outcomes in V^ω. It is unique whenever the arena is deterministic. We write $\mathsf{Out}(\sigma_1, \sigma_2)$ for that set and $\mathsf{Out}(\sigma_i)$ if only σ_i is fixed
- Winning condition for P_1: reach
- Winning condition for P_2: avoid
- Strategy σ_1 is winning whenever $\mathsf{Out}(\sigma_1) \subseteq V^*$

[AHK98,07] L. De Alfaro, T. Henzinger, O. Kupferman. Concurrent reachability games (FOCS'98, TCS'07)

Two-player (zero-sum) concurrent games



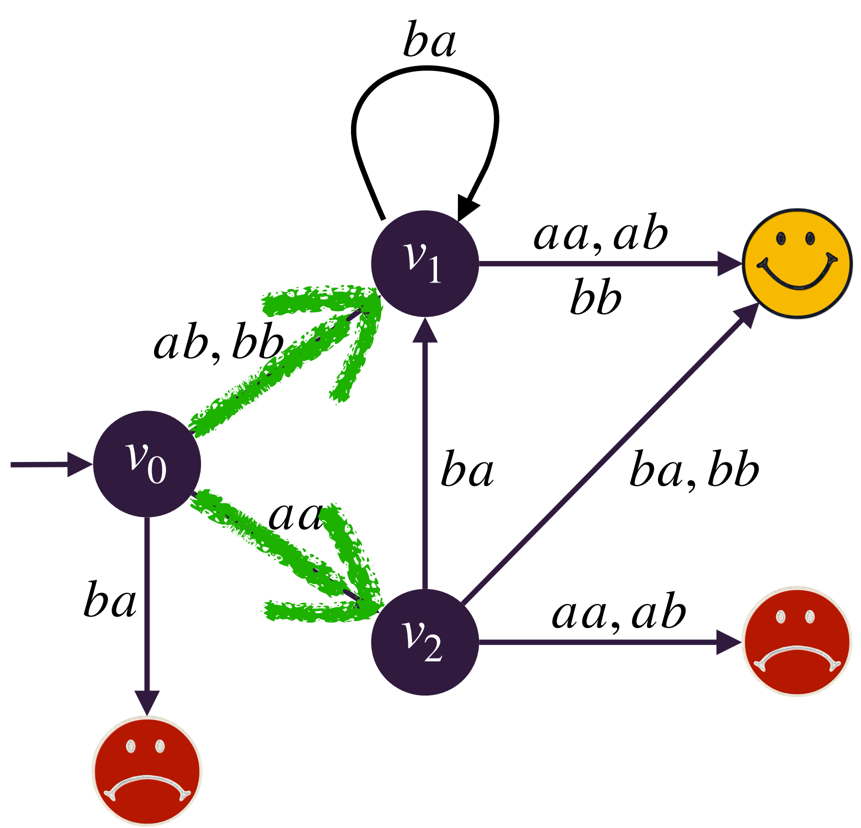
- Game graph $G = (V, \delta)$ with $\delta \subseteq V \times \Sigma^2 \times V$
- A strategy for player P_i is a mapping $\sigma_i : V^+ \to \Sigma$
- Given (σ_1, σ_2), there are possibly several outcomes in V^ω. It is unique whenever the arena is deterministic. We write $\mathsf{Out}(\sigma_1, \sigma_2)$ for that set and $\mathsf{Out}(\sigma_i)$ if only σ_i is fixed
- Winning condition for P_1: reach
- Winning condition for P_2: avoid
- Strategy σ_1 is winning whenever $\mathsf{Out}(\sigma_1) \subseteq V^*$

Example of winning strategy for P_1:

$$\sigma_1(v_0) = a,\ \sigma_1(v_0 v_1) = a,\ \sigma_1(v_0 v_2) = b,\ \sigma_1(v_0 v_2 v_1) = a$$

[AHK98,07] L. De Alfaro, T. Henzinger, O. Kupferman. Concurrent reachability games (FOCS'98, TCS'07)

Two-player (zero-sum) concurrent games



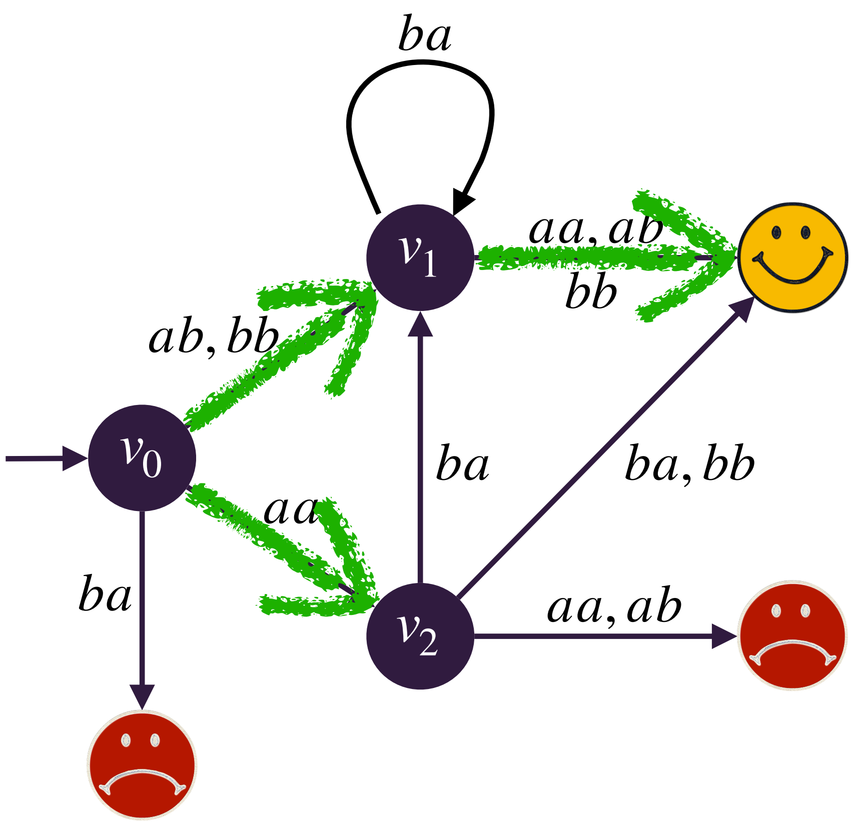
- Game graph $G = (V, \delta)$ with $\delta \subseteq V \times \Sigma^2 \times V$
- A strategy for player P_i is a mapping $\sigma_i : V^+ \to \Sigma$
- Given (σ_1, σ_2), there are possibly several outcomes in V^ω. It is unique whenever the arena is deterministic. We write $\mathsf{Out}(\sigma_1, \sigma_2)$ for that set and $\mathsf{Out}(\sigma_i)$ if only σ_i is fixed
- Winning condition for P_1: reach
- Winning condition for P_2: avoid
- Strategy σ_1 is winning whenever $\mathsf{Out}(\sigma_1) \subseteq V^*$

Example of winning strategy for P_1:

$\sigma_1(v_0) = a$, $\sigma_1(v_0v_1) = a$, $\sigma_1(v_0v_2) = b$, $\sigma_1(v_0v_2v_1) = a$

[AHK98,07] L. De Alfaro, T. Henzinger, O. Kupferman. Concurrent reachability games (FOCS'98, TCS'07)

Two-player (zero-sum) concurrent games



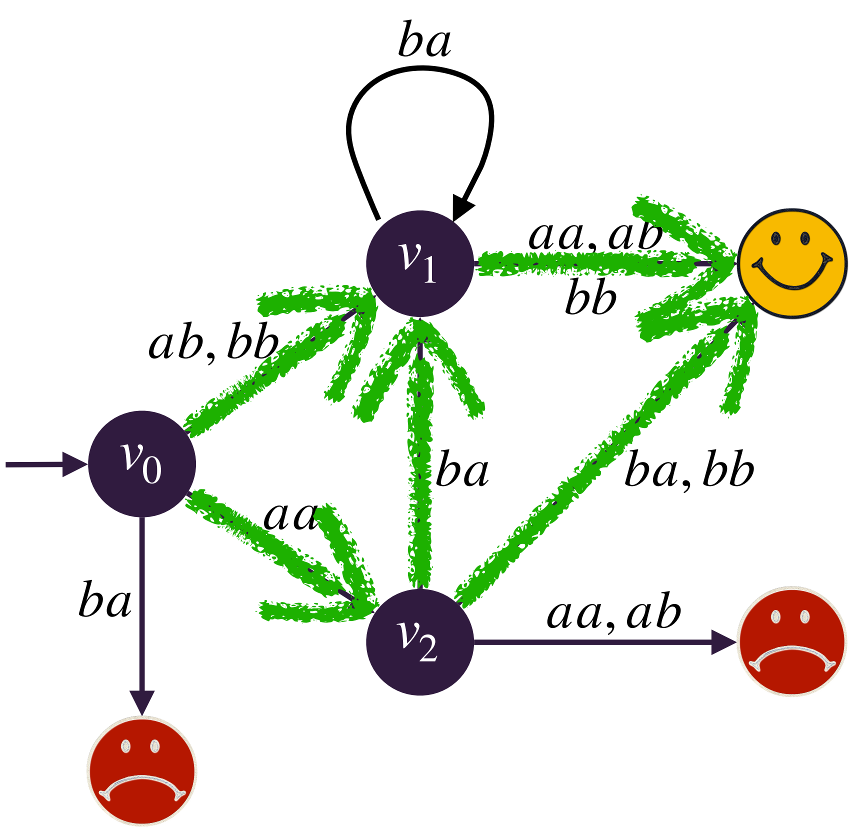
- Game graph $G = (V, \delta)$ with $\delta \subseteq V \times \Sigma^2 \times V$
- A strategy for player P_i is a mapping $\sigma_i : V^+ \to \Sigma$
- Given (σ_1, σ_2), there are possibly several outcomes in V^ω. It is unique whenever the arena is deterministic. We write $\mathsf{Out}(\sigma_1, \sigma_2)$ for that set and $\mathsf{Out}(\sigma_i)$ if only σ_i is fixed
- Winning condition for P_1: reach
- Winning condition for P_2: avoid
- Strategy σ_1 is winning whenever $\mathsf{Out}(\sigma_1) \subseteq V^*$

Example of winning strategy for P_1:

$\sigma_1(v_0) = a$, $\sigma_1(v_0v_1) = a$, $\sigma_1(v_0v_2) = b$, $\sigma_1(v_0v_2v_1) = a$

[AHK98,07] L. De Alfaro, T. Henzinger, O. Kupferman. Concurrent reachability games (FOCS'98, TCS'07)

Two-player (zero-sum) concurrent games



- Game graph $G = (V, \delta)$ with $\delta \subseteq V \times \Sigma^2 \times V$
- A strategy for player P_i is a mapping $\sigma_i : V^+ \to \Sigma$
- Given (σ_1, σ_2), there are possibly several outcomes in V^ω. It is unique whenever the arena is deterministic. We write $\mathsf{Out}(\sigma_1, \sigma_2)$ for that set and $\mathsf{Out}(\sigma_i)$ if only σ_i is fixed
- Winning condition for P_1: reach 🙂
- Winning condition for P_2: avoid 🙂
- Strategy σ_1 is winning whenever $\mathsf{Out}(\sigma_1) \subseteq V^*$ 🙂

Example of winning strategy for P_1:

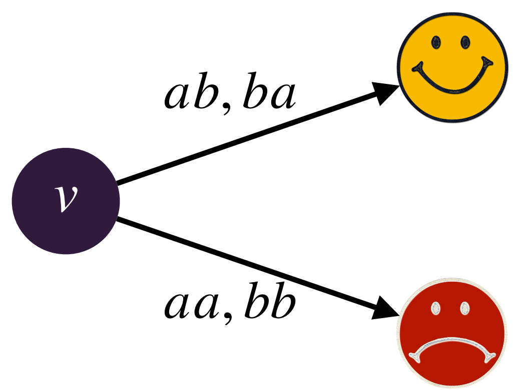
$\sigma_1(v_0) = a$, $\sigma_1(v_0 v_1) = a$, $\sigma_1(v_0 v_2) = b$, $\sigma_1(v_0 v_2 v_1) = a$

[AHK98,07] L. De Alfaro, T. Henzinger, O. Kupferman. Concurrent reachability games (FOCS'98, TCS'07)

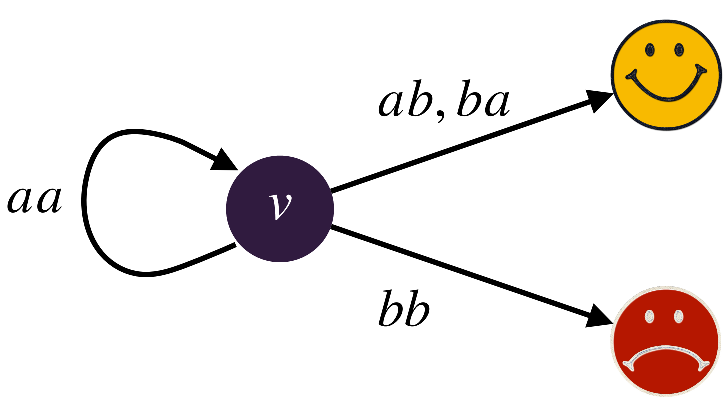
What do we know about those games?

- For many objectives, one can compute winning states and (deterministic) winning strategies for each of the players

- Those games are not determined with deterministic strategies



- They nevertheless have values and almost-optimal winning strategies for both players (Martin's second determinacy results for Blackwell games)



$val_1 = 1$

$\sigma_1(v) = (1 - \varepsilon) \cdot a + \varepsilon \cdot b$ is an ε-optimal strategy

No optimal strategy exists

[Mar98] D. A. Martin. The determinacy of Blackwell games (The Journal of Symbolic Logic'98)

From two-player to N-player concurrent games

From two-player to N-player concurrent games

Two players:

$$v_0 \xrightarrow{ab} v_1$$

with $a, b \in \Sigma$

From two-player to N-player concurrent games

Two players:

$$v_0 \xrightarrow{ab} v_1$$

with $a, b \in \Sigma$

N players:

$$v_0 \xrightarrow{w} v_1$$

with $w \in \Sigma^N$

From two-player to N-player concurrent games

Two players:

$$v_0 \xrightarrow{ab} v_1$$

with $a, b \in \Sigma$

N players:

$$v_0 \xrightarrow{w} v_1$$

with $w \in \Sigma^N$

- Game graph $G = (V, \delta)$ with $\delta \subseteq V \times \Sigma^N \times V$
- A strategy for player P_i is a mapping $\sigma_i : V^+ \to \Sigma$
- Given $\sigma = (\sigma_i)_{1 \leq i \leq N}$, there are possibly several outcomes in V^ω. It is unique whenever the arena is deterministic. We write $\mathsf{Out}(\sigma)$ for that set and $\mathsf{Out}(\sigma_i)$ if only σ_i is fixed

From two-player to N-player concurrent games

Two players:

$$v_0 \xrightarrow{ab} v_1$$

with $a, b \in \Sigma$

N players:

$$v_0 \xrightarrow{w} v_1$$

with $w \in \Sigma^N$

- Game graph $G = (V, \delta)$ with $\delta \subseteq V \times \Sigma^N \times V$
- A strategy for player P_i is a mapping $\sigma_i : V^+ \to \Sigma$
- Given $\sigma = (\sigma_i)_{1 \leq i \leq N}$, there are possibly several outcomes in V^ω. It is unique whenever the arena is deterministic. We write $\mathsf{Out}(\sigma)$ for that set and $\mathsf{Out}(\sigma_i)$ if only σ_i is fixed

- Use of these games:
  - For coordination (specific Nature player, and partial observability) *[DMV18]* — linked to distributed synthesis *[MW03]*
  - For rational synthesis (e.g. constrained Nash equilibria) *[BBNM15,KPV16]*

[MW03] S. Mohalik, I. Walukiewicz. Distributed Games (FSTTCS'03)
[BBMU15] P. Bouyer, R. Brenguier, N. Markey, M. Ummels. Pure Nash Equilibria in Concurrent Deterministic Games (LMCS'15)
[KPV16] O. Kupferman, G. Perelli, M. Vardi. Synthesis with Rational Environments (AMAI'16)
[DMV18] D. Berwanger, A.B. Mathew, M. van den Bogaard. Hierarchical Information and the Synthesis of Distributed Strategies (Acta Informatica'18)

Parameterized verification

Parameterized verification

- **Standard verification:** can only verify instances of the system, where the value of the parameter is known

Fix N, and check $S(N) \models \varphi$

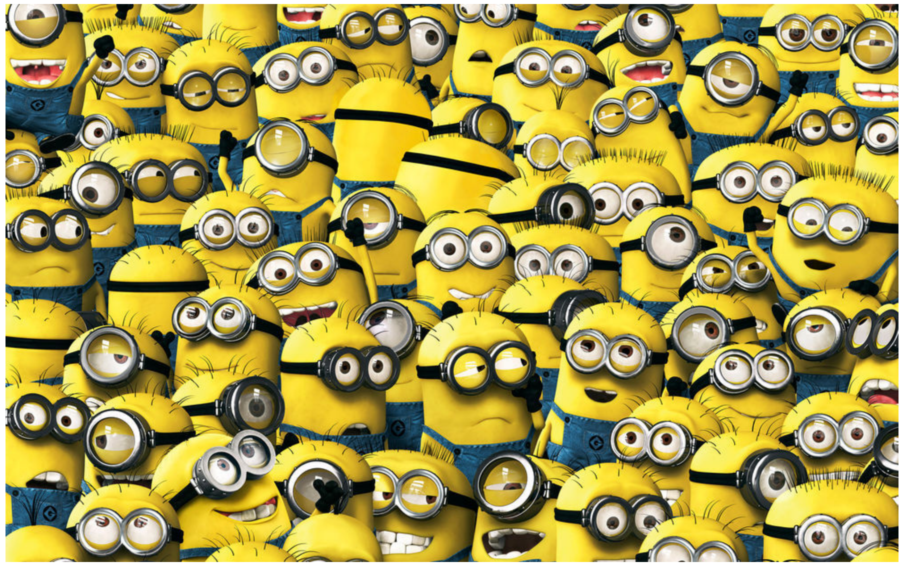
Parameterized verification

- **Standard verification:** can only verify instances of the system, where the value of the parameter is known

Fix N, and check $S(N) \models \varphi$

- **Parameterized verification:** design algorithms to verify all instances of the system, at once

Check that $S(N) \models \varphi$ for every N



Parameterized verification

- **Standard verification:** can only verify instances of the system, where the value of the parameter is known

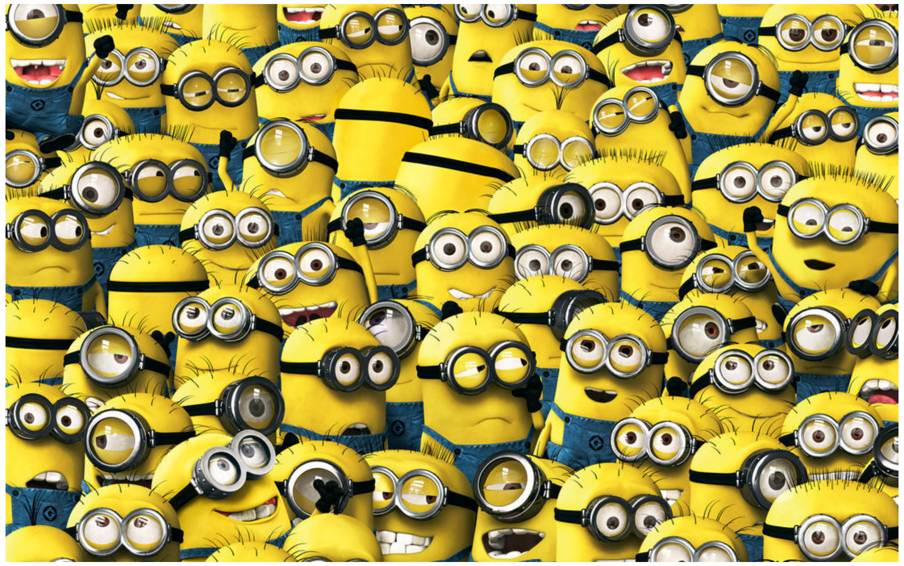
$$\text{Fix } N\text{, and check } S(N) \models \varphi$$

- **Parameterized verification:** design algorithms to verify all instances of the system, at once

$$\text{Check that } S(N) \models \varphi \text{ for every } N$$

Various kinds of parameters:

- In arithmetic contraints (timed automata, counter automata, hybrid automata, Markov chains, ...)
- Number of agents



Why parameterized verification?

Why parameterized verification?

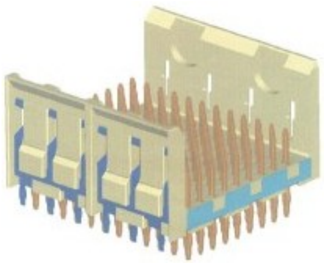
- Abstraction for systems with an arbitrary or unknown number of participants

Why parameterized verification?

- Abstraction for systems with an arbitrary or unknown number of participants
- Examples:
  - Distributed algorithms (e.g. leader election protocol)
  - Network protocols
  - Swarm intelligence systems
  - ...

Why parameterized verification?

- Abstraction for systems with an arbitrary or unknown number of participants
- Examples:
  - Distributed algorithms (e.g. leader election protocol)
  - Network protocols
  - Swarm intelligence systems
  - ...

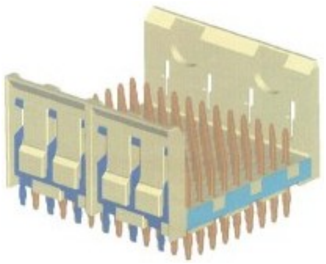


- It is not true that errors always occur with small instances of the parameters
  - Example of the Futurebus+ cache coherence protocol

[BJK+15] R. Bloem, S. Jacobs, A. Khalimov, I. Konnov, S. Rubin, H. Veith, J. Widder: Decidability of Parameterized Verification

Why parameterized verification?

- Abstraction for systems with an arbitrary or unknown number of participants
- Examples:
  - Distributed algorithms (e.g. leader election protocol)
  - Network protocols
  - Swarm intelligence systems
  - ...



- It is not true that errors always occur with small instances of the parameters
  - Example of the Futurebus+ cache coherence protocol
- Need to design methods for verifying parameterized systems, not only instances

[BJK+15] R. Bloem, S. Jacobs, A. Khalimov, I. Konnov, S. Rubin, H. Veith, J. Widder: Decidability of Parameterized Verification

Parameterized verification of crowds

[Esp14] J. Esparza. Keeping a Crowd Safe: On the Complexity of Parameterized Verification (STACS'14)
[BJK+15] R. Bloem, S. Jacobs, A. Khalimov, I. Konnov, S. Rubin, H. Veith, J. Widder: Decidability of Parameterized Verification

Parameterized verification of crowds

- Processes executing the same piece of code (crowd)

[Esp14] J. Esparza. Keeping a Crowd Safe: On the Complexity of Parameterized Verification (STACS'14)
[BJK+15] R. Bloem, S. Jacobs, A. Khalimov, I. Konnov, S. Rubin, H. Veith, J. Widder: Decidability of Parameterized Verification

Parameterized verification of crowds

- Processes executing the same piece of code (crowd)
- All decidability/complexity/memory issues depend on many features:
  - Communication structure (broadcasts, rendez-vous, shared variables, token-passing, …)
  - With or without fixed architecture
  - With or without identifiers
  - …

[Esp14] J. Esparza. Keeping a Crowd Safe: On the Complexity of Parameterized Verification (STACS'14)
[BJK+15] R. Bloem, S. Jacobs, A. Khalimov, I. Konnov, S. Rubin, H. Veith, J. Widder: Decidability of Parameterized Verification

Parameterized verification of crowds

- Processes executing the same piece of code (crowd)
- All decidability/complexity/memory issues depend on many features:
  - Communication structure (broadcasts, rendez-vous, shared variables, token-passing, …)
  - With or without fixed architecture
  - With or without identifiers
  - …
- Decidability often relies on:
  - Well-structured transition systems
  - Existence of cutoffs

[Esp14] J. Esparza. Keeping a Crowd Safe: On the Complexity of Parameterized Verification (STACS'14)
[BJK+15] R. Bloem, S. Jacobs, A. Khalimov, I. Konnov, S. Rubin, H. Veith, J. Widder: Decidability of Parameterized Verification

From two-player to arbitrarily-many player concurrent games

From two-player to arbitrarily-many player concurrent games

Two players:

$$v_0 \xrightarrow{ab} v_1$$

with $a, b \in \Sigma$

N players:

$$v_0 \xrightarrow{w} v_1$$

with $w \in \Sigma^N$

From two-player to arbitrarily-many player concurrent games

Two players:

$$v_0 \xrightarrow{ab} v_1$$

with $a, b \in \Sigma$

N players:

$$v_0 \xrightarrow{w} v_1$$

with $w \in \Sigma^N$

Unknown number of players:

$$v_0 \xrightarrow{w} v_1$$

with $w \in \Sigma^+$

From two-player to arbitrarily-many player concurrent games

Two players:

$$v_0 \xrightarrow{ab} v_1$$

with $a, b \in \Sigma$

N players:

$$v_0 \xrightarrow{w} v_1$$

with $w \in \Sigma^N$

Unknown number of players:

$$v_0 \xrightarrow{w} v_1$$

with $w \in \Sigma^+$

- Game graph $G = (V, \delta)$ with $\delta \subseteq V \times \Sigma^+ \times V$

From two-player to arbitrarily-many player concurrent games

Two players:

$$v_0 \xrightarrow{ab} v_1$$

with $a, b \in \Sigma$

N players:

$$v_0 \xrightarrow{w} v_1$$

with $w \in \Sigma^N$

Unknown number of players:

$$v_0 \xrightarrow{w} v_1$$

with $w \in \Sigma^+$

- Game graph $G = (V, \delta)$ with $\delta \subseteq V \times \Sigma^+ \times V$
- A strategy for player P_i is a mapping $\sigma_i : V^+ \to \Sigma$
- Given $\sigma = (\sigma_i)_{1 \leq i \leq N}$, there are possibly several outcomes in V^ω. It is unique whenever the arena is deterministic. We write $\mathsf{Out}(\sigma)$ for that set and $\mathsf{Out}(\sigma_i)$ if only σ_i is fixed

From two-player to arbitrarily-many player concurrent games

Two players:

$$v_0 \xrightarrow{ab} v_1$$

with $a, b \in \Sigma$

N players:

$$v_0 \xrightarrow{w} v_1$$

with $w \in \Sigma^N$

Unknown number of players:

$$v_0 \xrightarrow{w} v_1$$

with $w \in \Sigma^+$

- Game graph $G = (V, \delta)$ with $\delta \subseteq V \times \Sigma^+ \times V$
- A strategy for player P_i is a mapping $\sigma_i : V^+ \to \Sigma$
- Given $\sigma = (\sigma_i)_{1 \leq i \leq N}$, there are possibly several outcomes in V^ω. It is unique whenever the arena is deterministic. We write $\mathsf{Out}(\sigma)$ for that set and $\mathsf{Out}(\sigma_i)$ if only σ_i is fixed

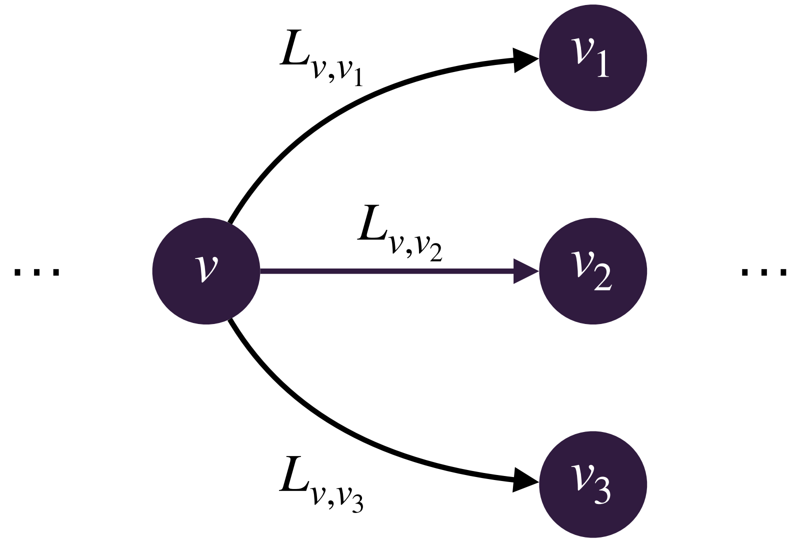
Assumption: for every $(v, v') \in V^2$,

$$\{w \in \Sigma^+ \mid (v, w, v') \in \delta\} \text{ is regular}$$

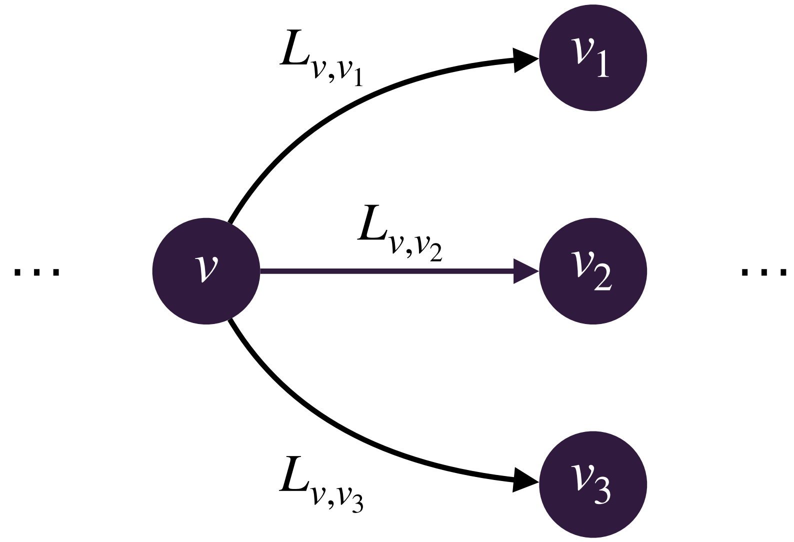
$$v_0 \xrightarrow{L} v_1$$

with L regular language

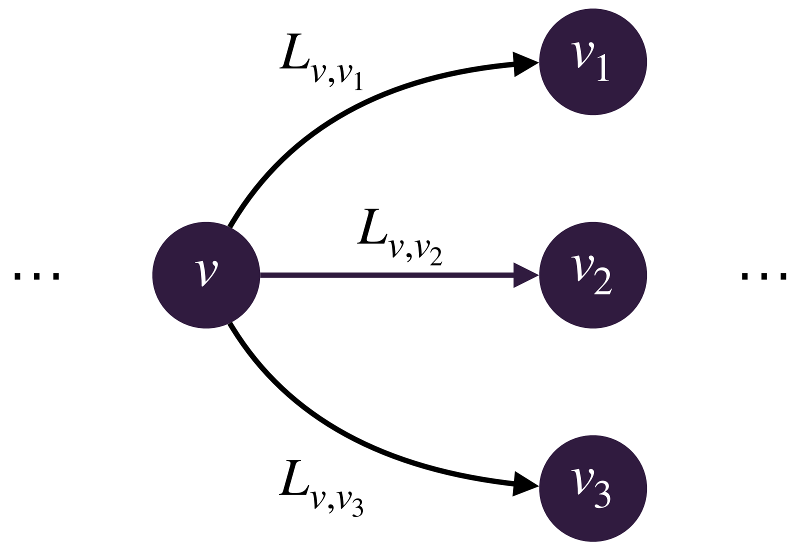
How do we play such a game?



How do we play such a game?

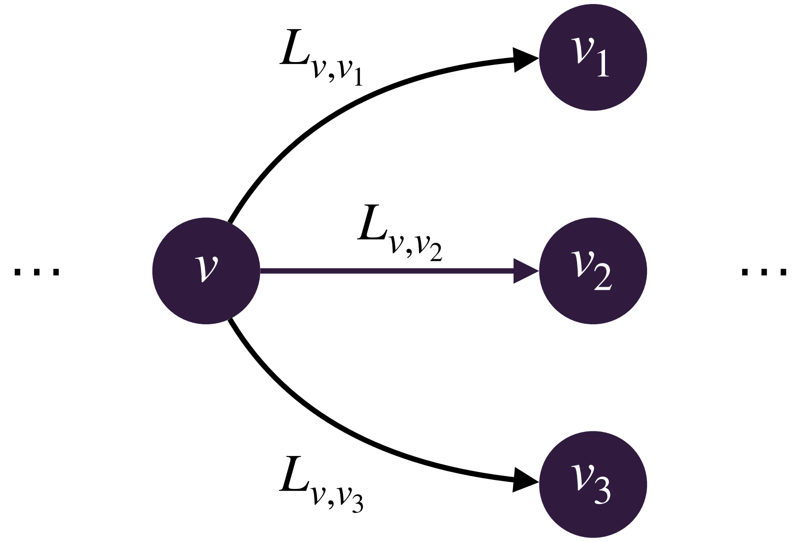


How do we play such a game?



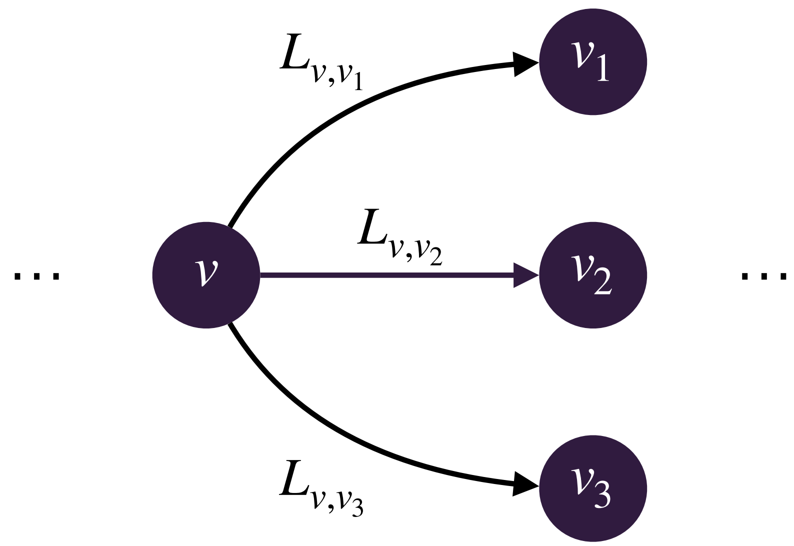
- A coalition of size k is chosen, but is unknown to the players

How do we play such a game?



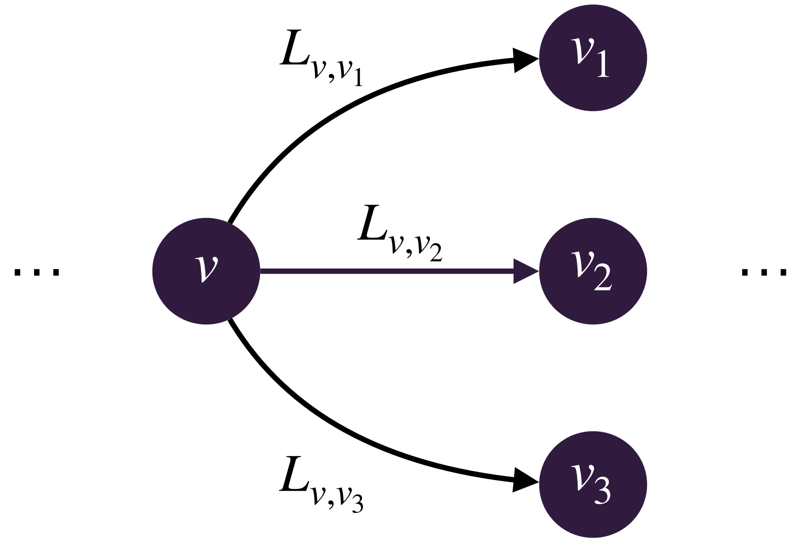
- A coalition of size k is chosen, but is unknown to the players
- Each player P_i knows she is the i-th (implicit identifier)

How do we play such a game?



- A coalition of size k is chosen, but is unknown to the players
- Each player P_i knows she is the i-th (implicit identifier)
- The game starts at initial vertex v_0

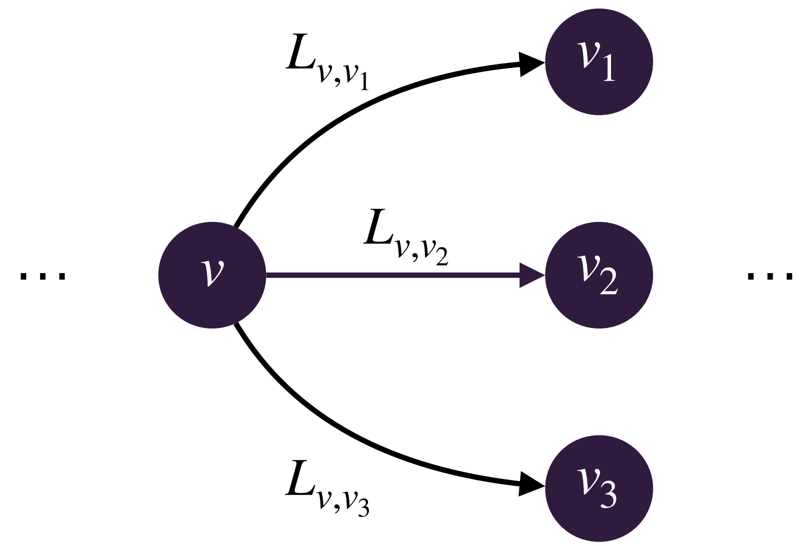
How do we play such a game?



- A coalition of size k is chosen, but is unknown to the players
- Each player P_i knows she is the i-th (implicit identifier)

- The game starts at initial vertex v_0
- Given the history (in V^*v) so far, each player P_i selects an action $a_i \in \Sigma$

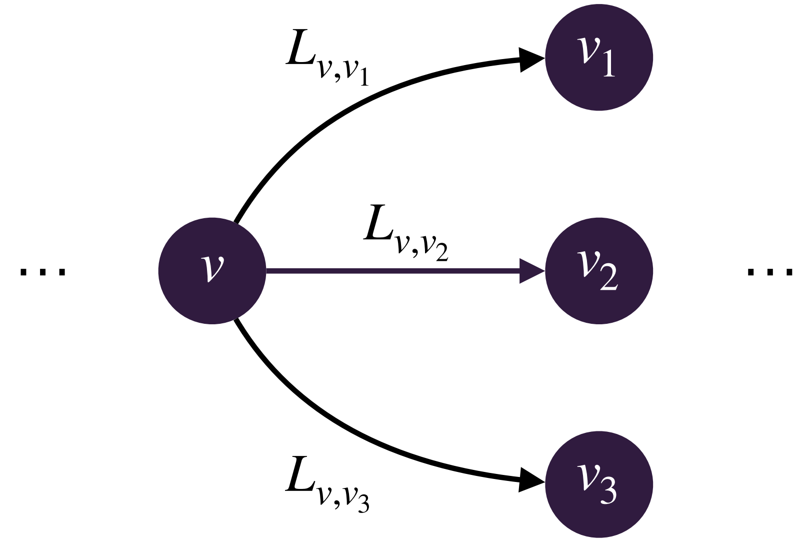
How do we play such a game?



- A coalition of size k is chosen, but is unknown to the players
- Each player P_i knows she is the i-th (implicit identifier)

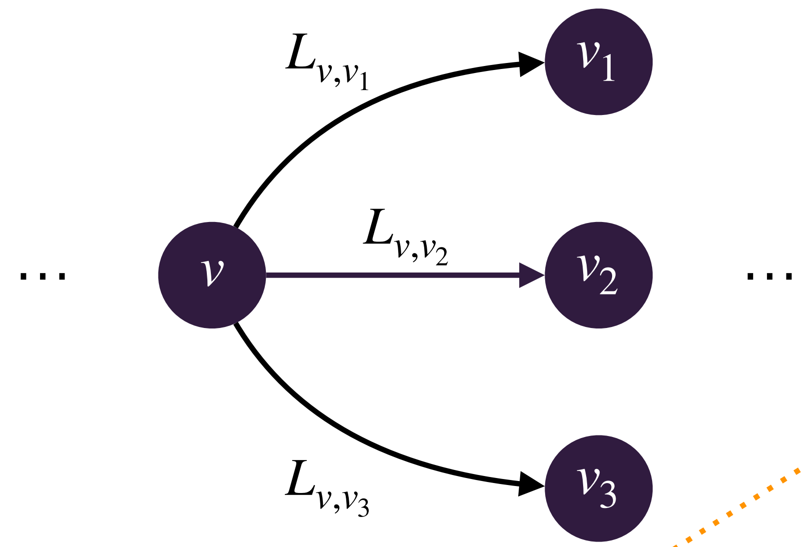
- The game starts at initial vertex v_0
- Given the history (in V^*v) so far, each player P_i selects an action $a_i \in \Sigma$
- The game proceeds to some v_j (non-det. choice) such that $a_1 a_2 \ldots a_k \in L_{v,v_j}$

How do we play such a game?



- A coalition of size k is chosen, but is unknown to the players
- Each player P_i knows she is the i-th (implicit identifier)
- The game starts at initial vertex v_0
- Given the history (in V^*v) so far, each player P_i selects an action $a_i \in \Sigma$
- The game proceeds to some v_j (non-det. choice) such that $a_1 a_2 \ldots a_k \in L_{v,v_j}$
- This produces an outcome in V^ω

How do we play such a game?



Adversarial choice

- A coalition of size k is chosen, but is unknown to the players
- Each player P_i knows she is the i-th (implicit identifier)
- The game starts at initial vertex v_0
- Given the history (in V^*v) so far, each player P_i selects an action $a_i \in \Sigma$
- The game proceeds to some v_j (non-det. choice) such that $a_1 a_2 \dots a_k \in L_{v,v_j}$
- This produces an outcome in V^ω

Adversarial choice

Two synthesis problems

Two synthesis problems

The crowd controller problem

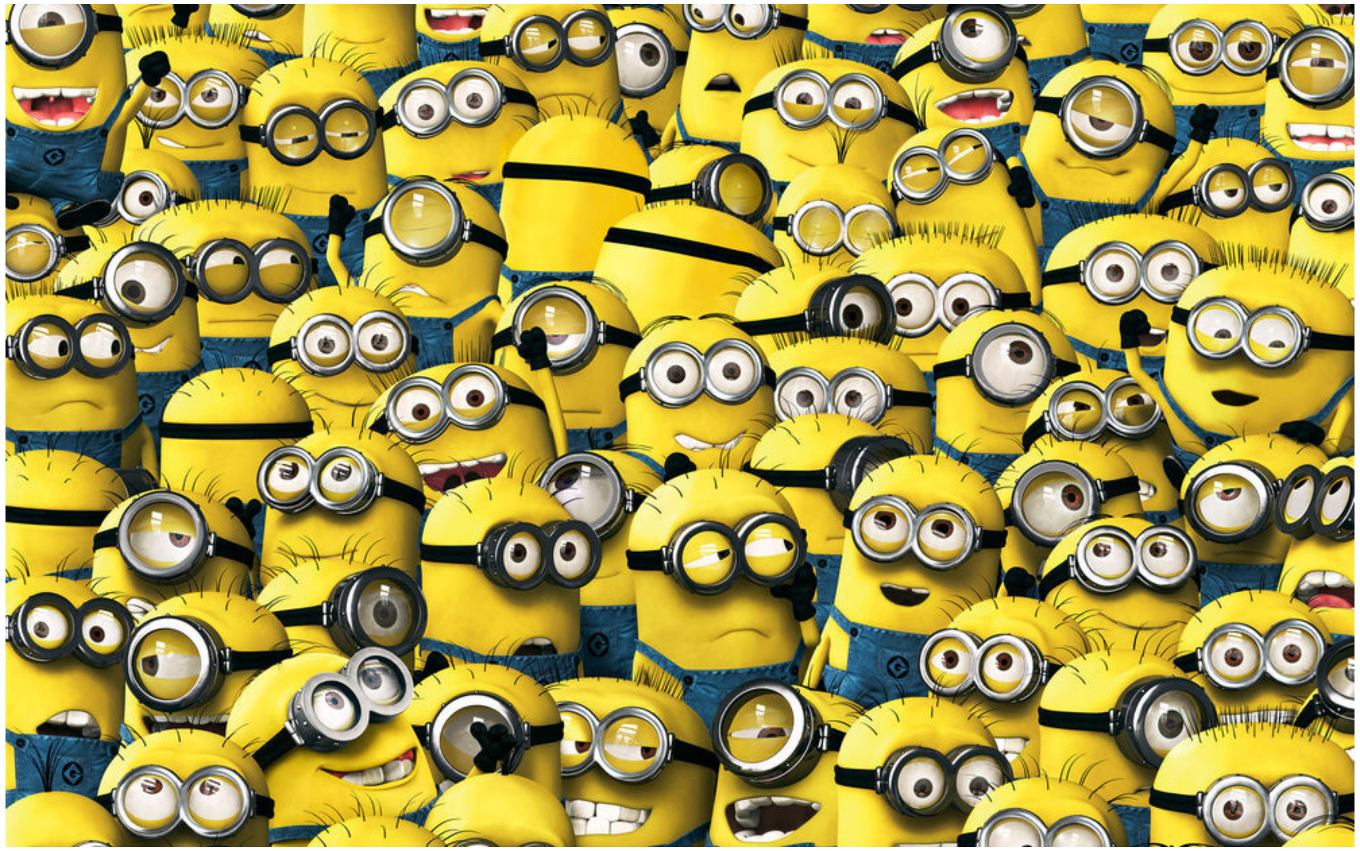
« Gru wants to guide/control the Minions »

Two synthesis problems

The crowd controller problem

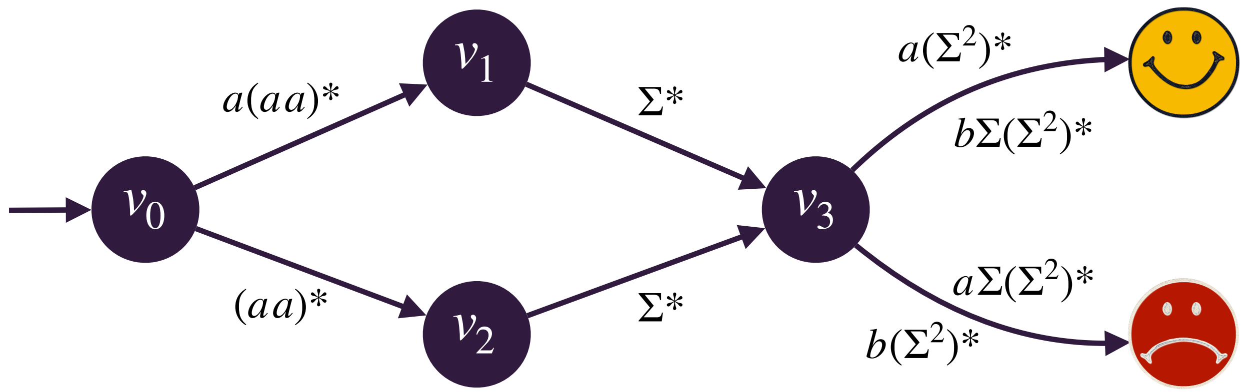
« Gru wants to guide/control the Minions »

The coalition synthesis problem

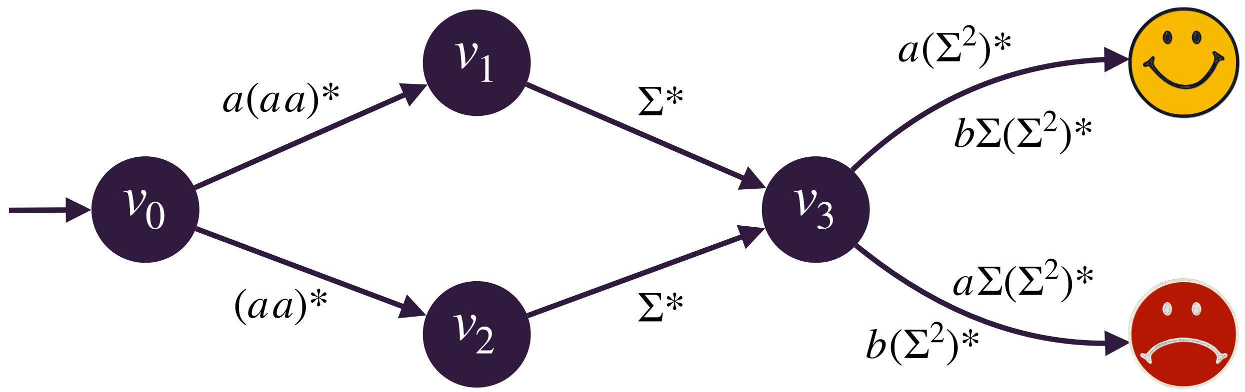


« The Minions want to achieve some goal »

The crowd controller problem

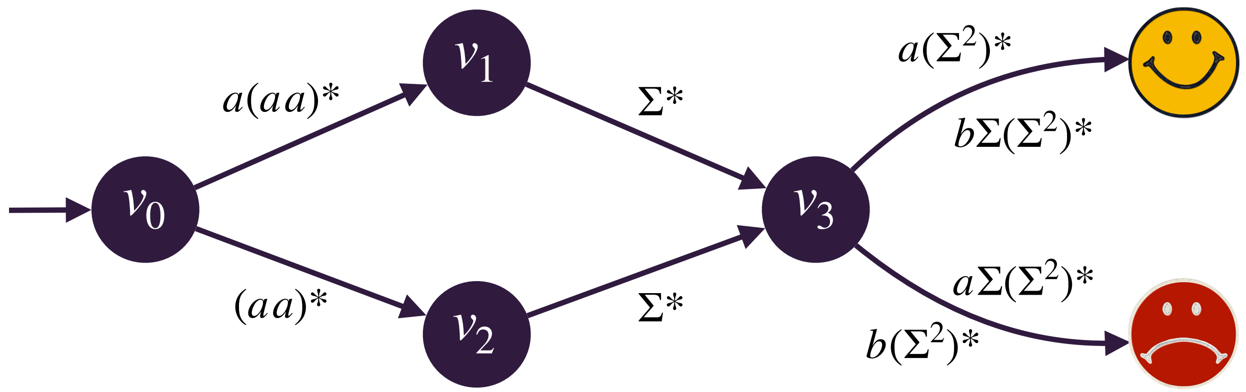


The crowd controller problem



Can player P_1 enforce ?

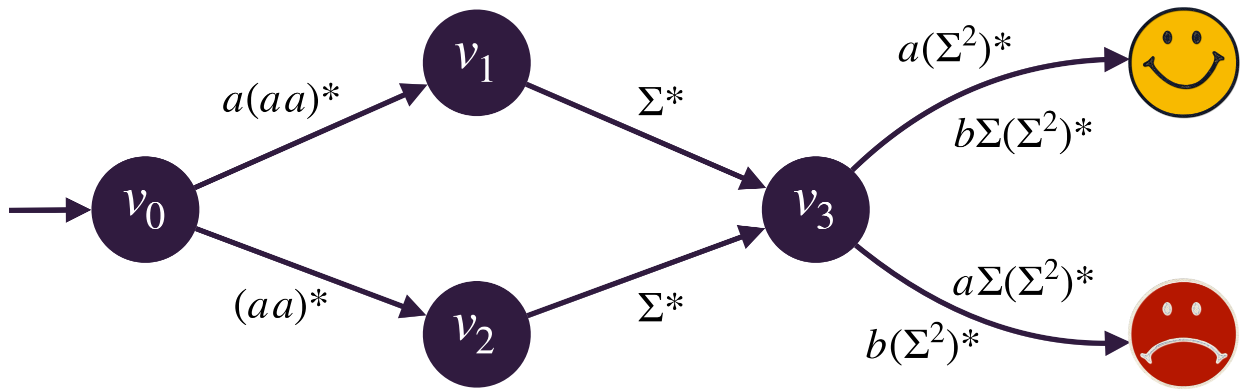
The crowd controller problem



Can player P_1 enforce 🙂 ?

- The number k of players is chosen (but unknown to everyone)

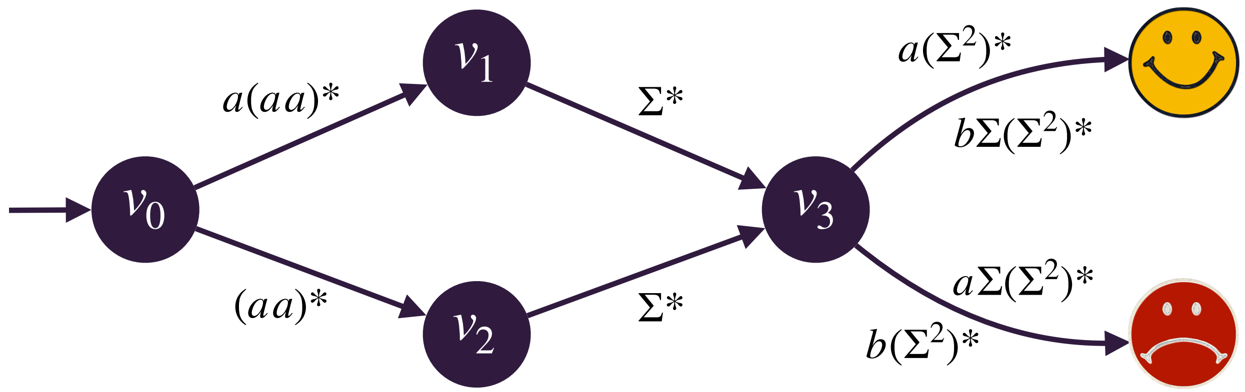
The crowd controller problem



Can player P_1 enforce ?

- The number k of players is chosen (but unknown to everyone)
- Assume the following strategy for player P_1:

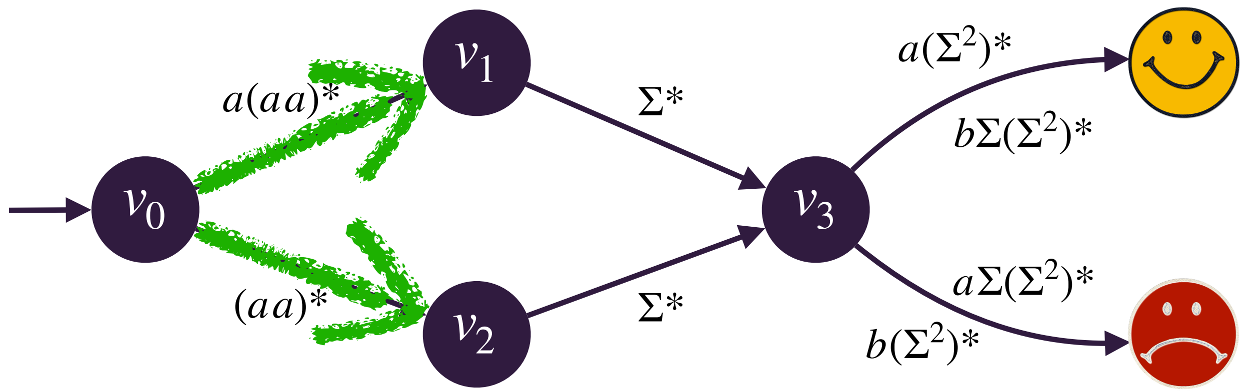
The crowd controller problem



Can player P_1 enforce ?

- The number k of players is chosen (but unknown to everyone)
- Assume the following strategy for player P_1:
  - $\sigma_1(v_0) = \sigma_1(v_0 v_1) = \sigma_1(v_0 v_2) = a$
  - $\sigma_1(v_0 v_1 v_3) = a$
  - $\sigma_1(v_0 v_2 v_3) = b$

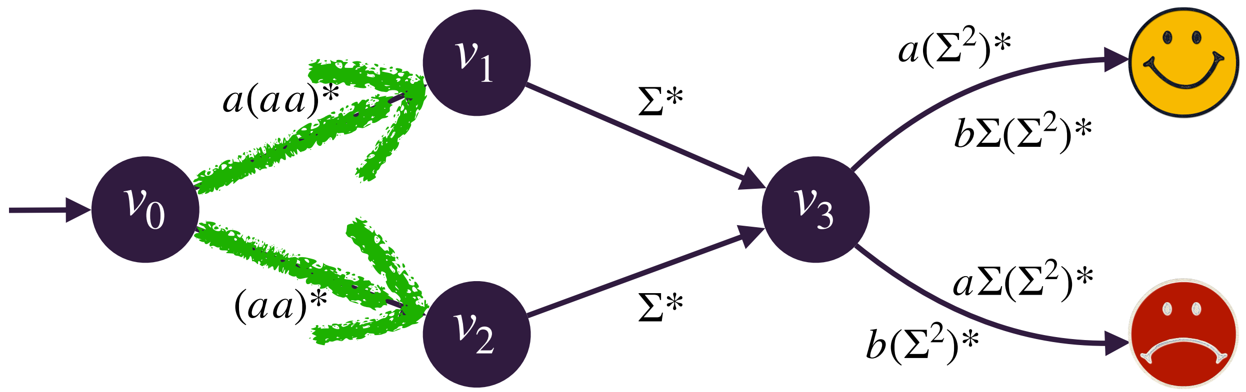
The crowd controller problem



Can player P_1 enforce 🙂 ?

- The number k of players is chosen (but unknown to everyone)
- Assume the following strategy for player P_1:
  - $\sigma_1(v_0) = \sigma_1(v_0v_1) = \sigma_1(v_0v_2) = a$
  - $\sigma_1(v_0v_1v_3) = a$
  - $\sigma_1(v_0v_2v_3) = b$

The crowd controller problem

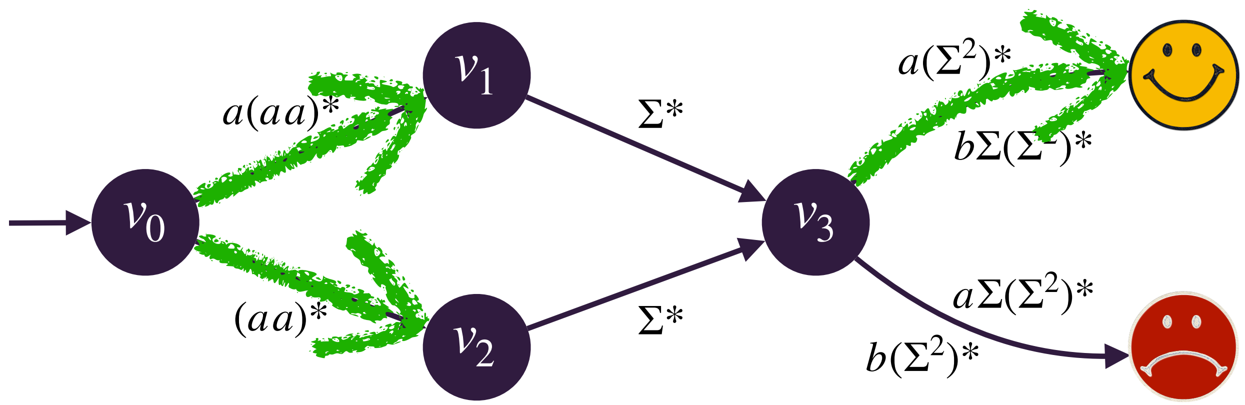


Can player P_1 enforce 🙂 ?

- The number k of players is chosen (but unknown to everyone)
- Assume the following strategy for player P_1:
  - $\sigma_1(v_0) = \sigma_1(v_0 v_1) = \sigma_1(v_0 v_2) = a$
  - $\sigma_1(v_0 v_1 v_3) = a$
  - $\sigma_1(v_0 v_2 v_3) = b$
- If k is odd, the game proceeds to v_1
  If k is even, the game proceeds to v_2

(P_1 learns it when visiting v_1 or v_2)

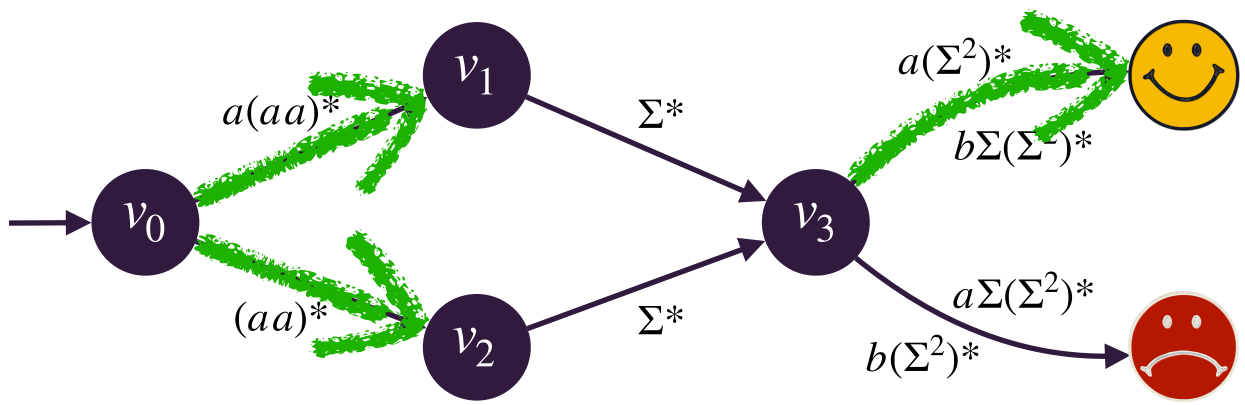
The crowd controller problem



Can player P_1 enforce ☺?

- The number k of players is chosen (but unknown to everyone)
- Assume the following strategy for player P_1:
  - $\sigma_1(v_0) = \sigma_1(v_0 v_1) = \sigma_1(v_0 v_2) = a$
  - $\sigma_1(v_0 v_1 v_3) = a$
  - $\sigma_1(v_0 v_2 v_3) = b$
- If k is odd, the game proceeds to v_1
  If k is even, the game proceeds to v_2 (P_1 learns it when visiting v_1 or v_2)

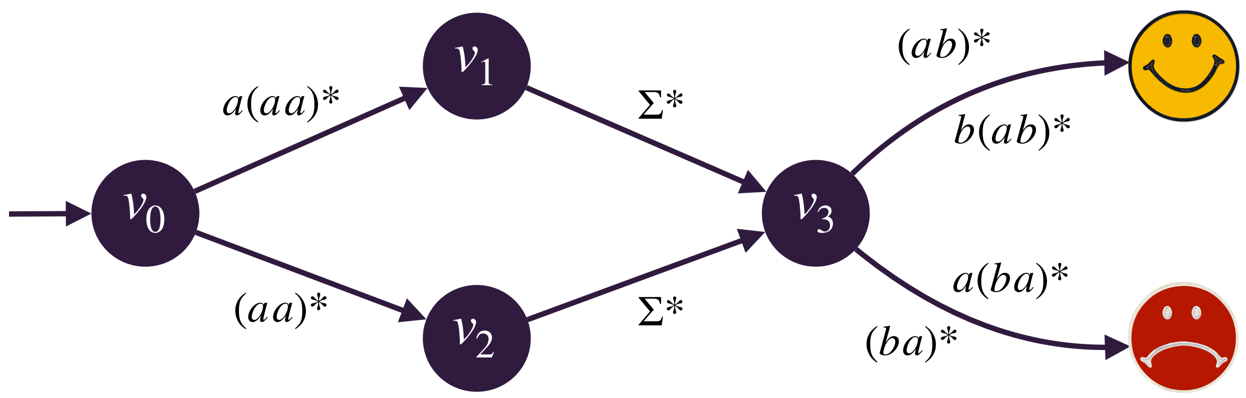
The crowd controller problem

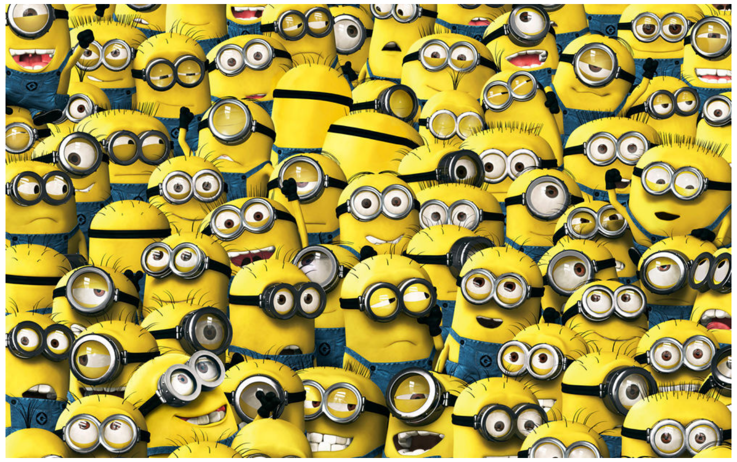


Can player P_1 enforce ?

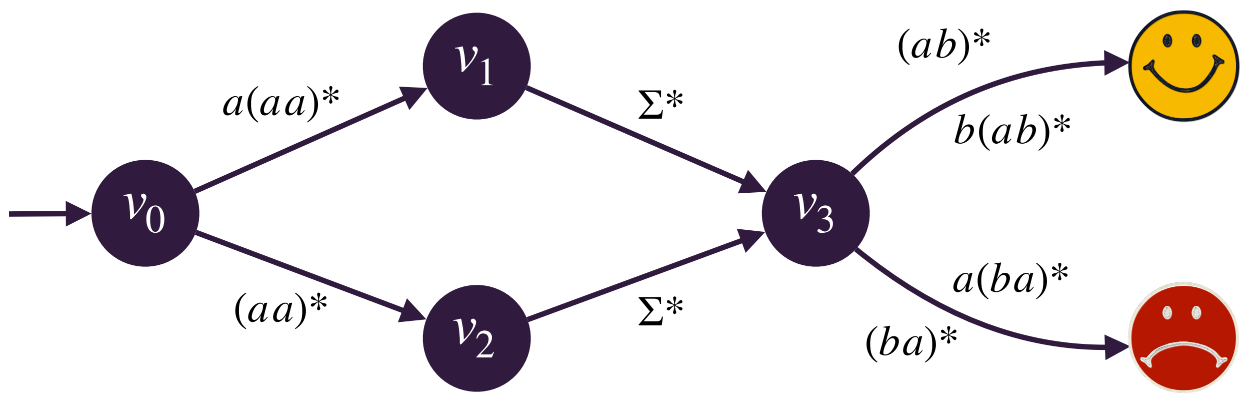
- The number k of players is chosen (but unknown to everyone)
- Assume the following strategy for player P_1:
  - $\sigma_1(v_0) = \sigma_1(v_0v_1) = \sigma_1(v_0v_2) = a$
  - $\sigma_1(v_0v_1v_3) = a$
  - $\sigma_1(v_0v_2v_3) = b$
- If k is odd, the game proceeds to v_1
  If k is even, the game proceeds to v_2 (P_1 learns it when visiting v_1 or v_2)
- In both cases, the choice of P_1 at v_3 ensures reaching

The coalition problem

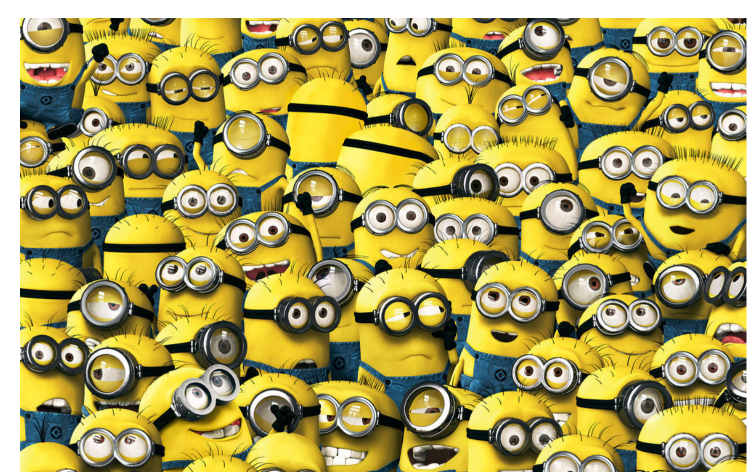




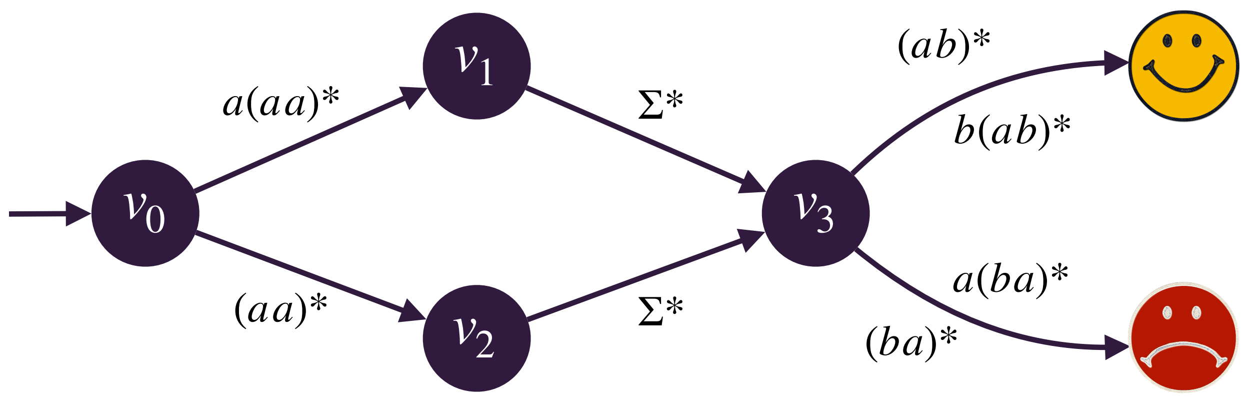
The coalition problem



Can any coalition ensure ?



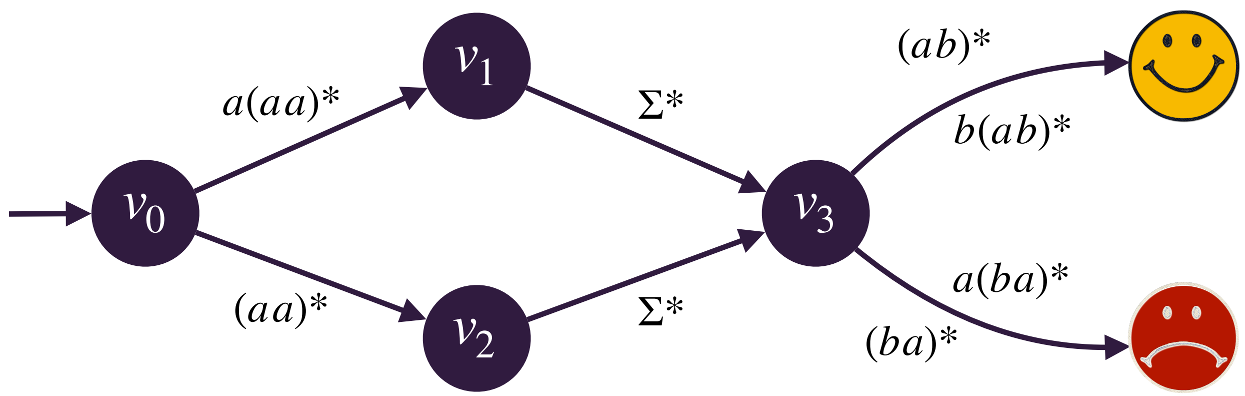
The coalition problem



Can any coalition ensure ?

- The number k of players is chosen (but unknown to everyone)

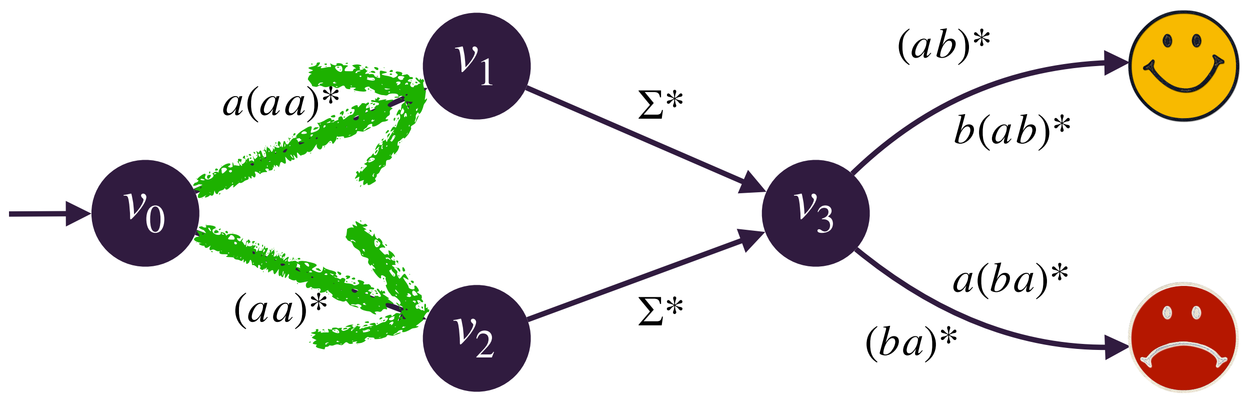
The coalition problem



Can any coalition ensure ?

- The number k of players is chosen (but unknown to everyone)
- At v_0, each player chooses a

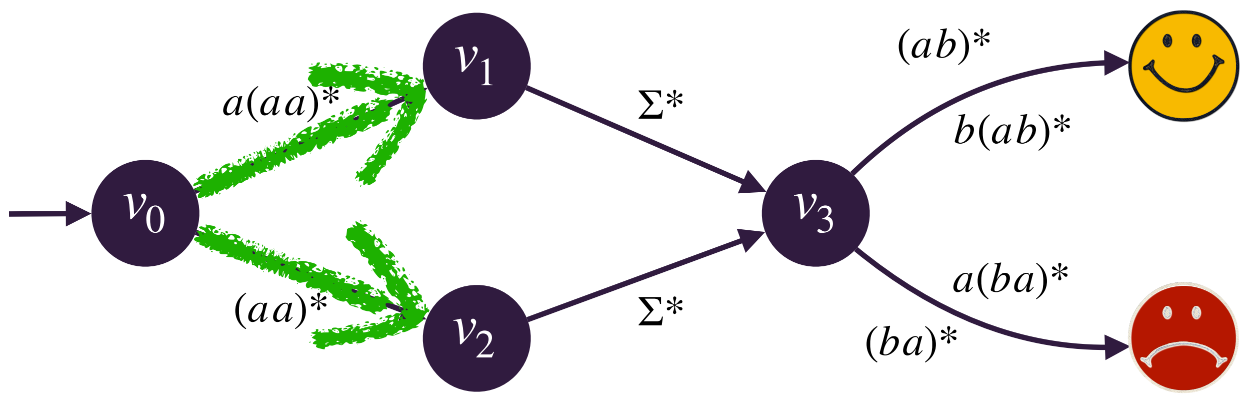
The coalition problem



Can any coalition ensure 🙂?

- The number k of players is chosen (but unknown to everyone)
- At v_0, each player chooses a

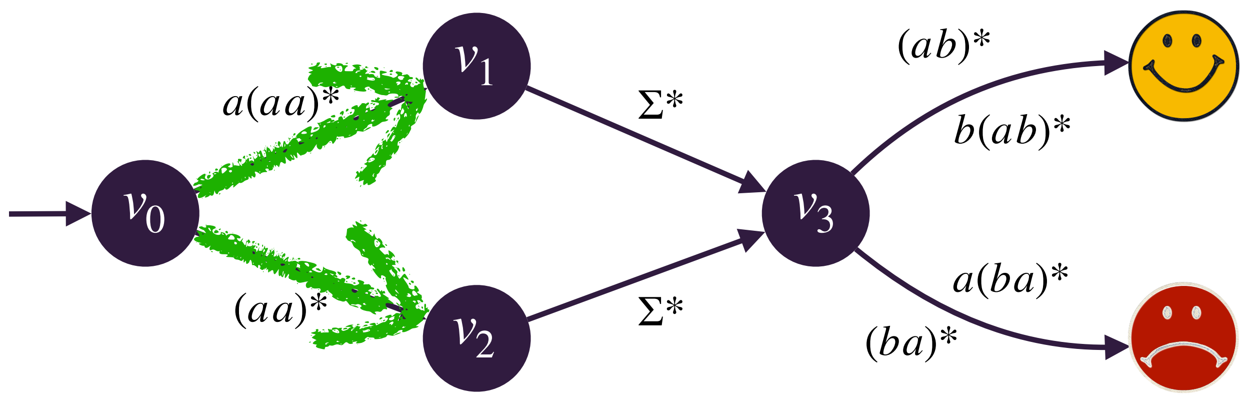
The coalition problem



Can any coalition ensure 🙂?

- The number k of players is chosen (but unknown to everyone)
- At v_0, each player chooses a
- If k is odd, the game proceeds to v_1 (all players learn it when visiting v_1 or v_2)
  If k is even, the game proceeds to v_2

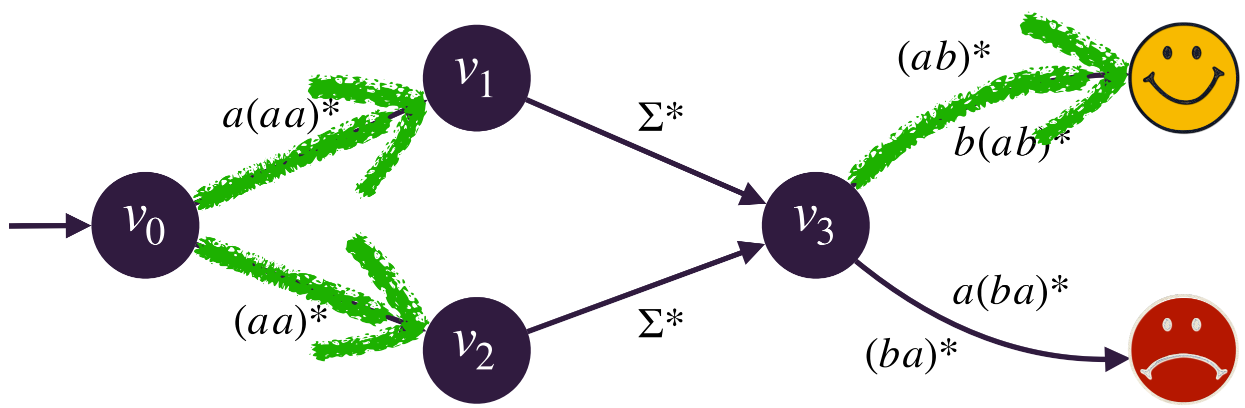
The coalition problem



Can any coalition ensure 🙂?

- The number k of players is chosen (but unknown to everyone)
- At v_0, each player chooses a
- If k is odd, the game proceeds to v_1 (all players learn it when visiting v_1 or v_2)
  If k is even, the game proceeds to v_2
- For each i:
  - $\sigma_{2i}(v_0v_1v_3) = a$ and $\sigma_{2i}(v_0v_2v_3) = b$
  - $\sigma_{2i+1}(v_0v_1v_3) = b$ and $\sigma_{2i+1}(v_0v_2v_3) = a$

The coalition problem



Can any coalition ensure 🙂?

- The number k of players is chosen (but unknown to everyone)
- At v_0, each player chooses a
- If k is odd, the game proceeds to v_1 (all players learn it when visiting v_1 or v_2)
  If k is even, the game proceeds to v_2
- For each i:
  - $\sigma_{2i}(v_0v_1v_3) = a$ and $\sigma_{2i}(v_0v_2v_3) = b$
  - $\sigma_{2i+1}(v_0v_1v_3) = b$ and $\sigma_{2i+1}(v_0v_2v_3) = a$
- For every k, $\mathsf{Out}((\sigma_i)_{1 \le i \le k}) \subseteq V^*$ 🙂

The crowd controller problem

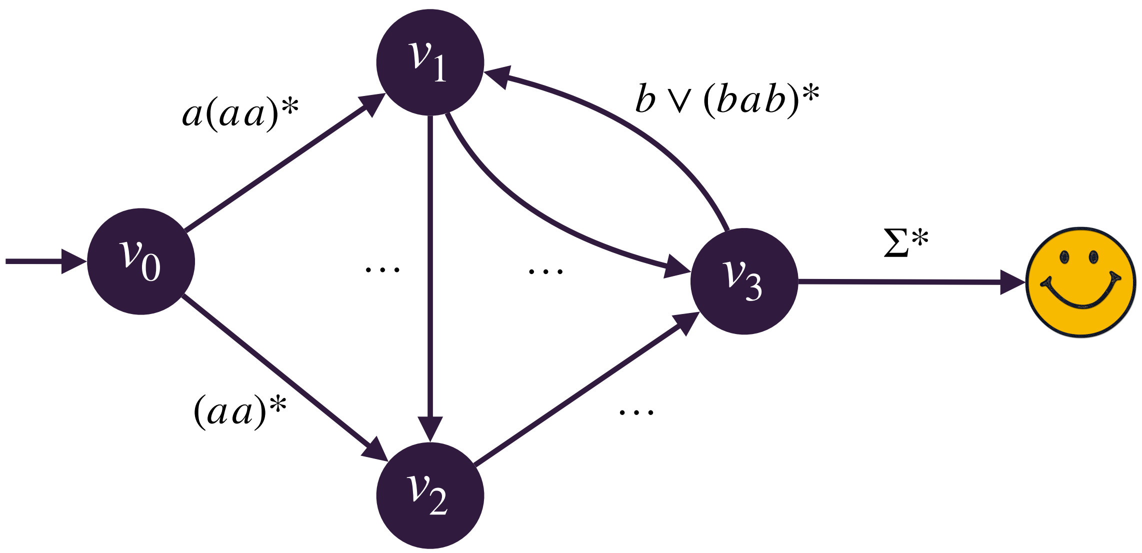
The crowd controller problem

- Input: parameterized game $G = (V, \delta)$ and linear property φ
- Question: does there exist σ_1 s.t. for every k, for every $(\sigma_i)_{2\leq i\leq k}$, for every $\rho \in \mathsf{Out}\big((\sigma_i)_{1\leq i\leq k}\big)$, $\rho \models \varphi$?

P_1 = Gru

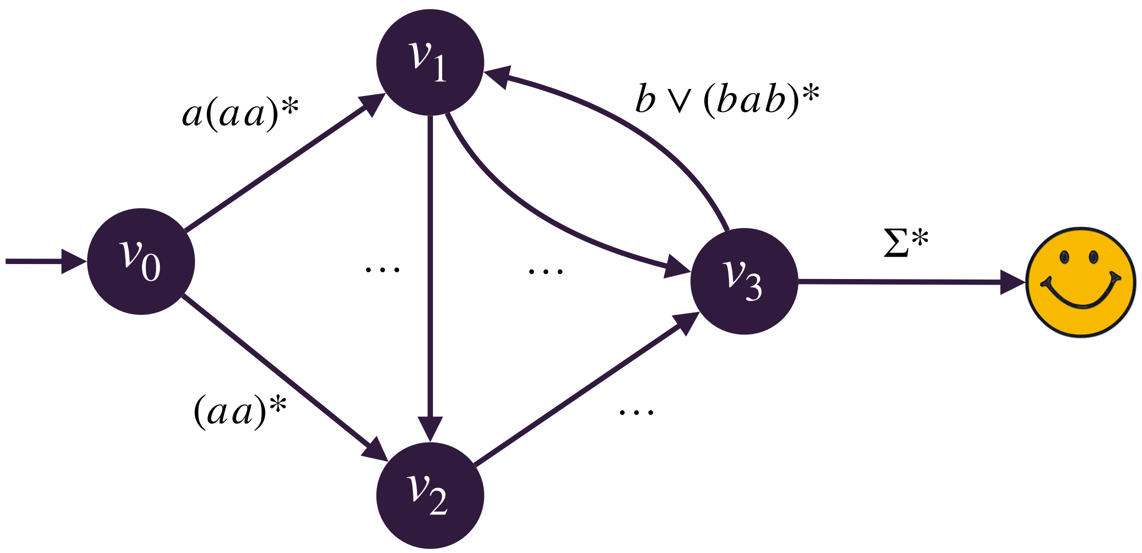
From languages to counting

φ objective



From languages to counting

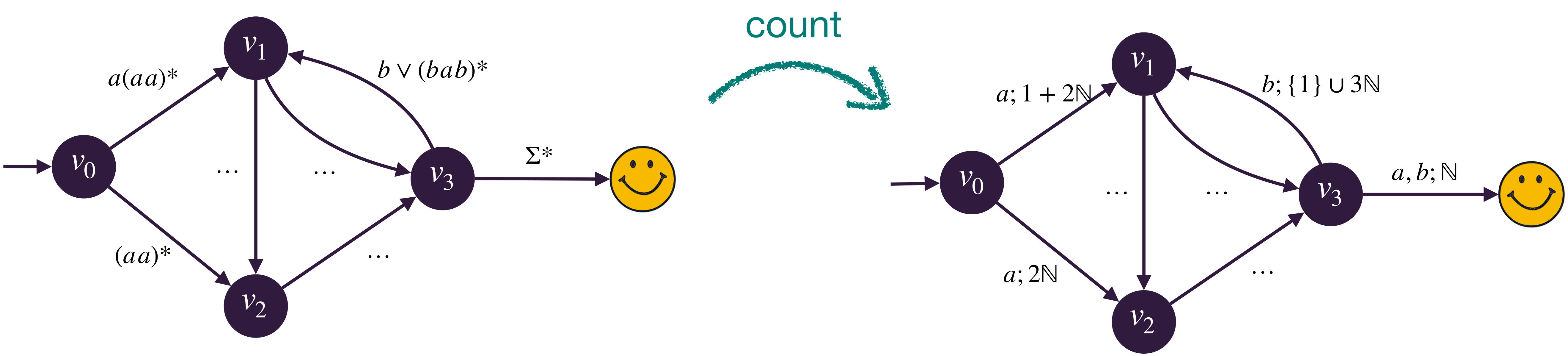
φ objective



count

From languages to counting

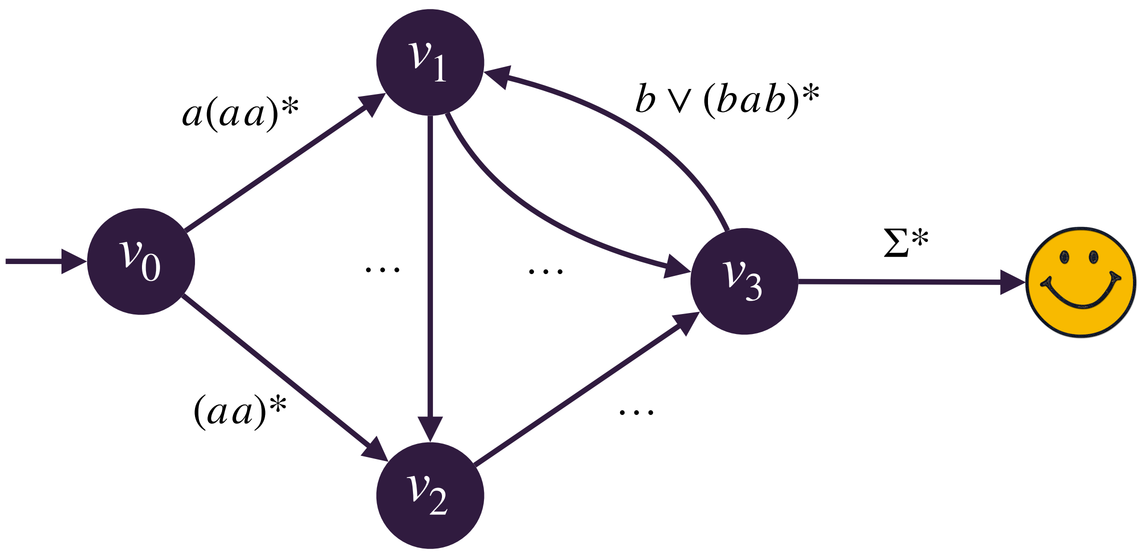
φ objective φ objective



Note : L regular language implies count(L) is a semi linear set

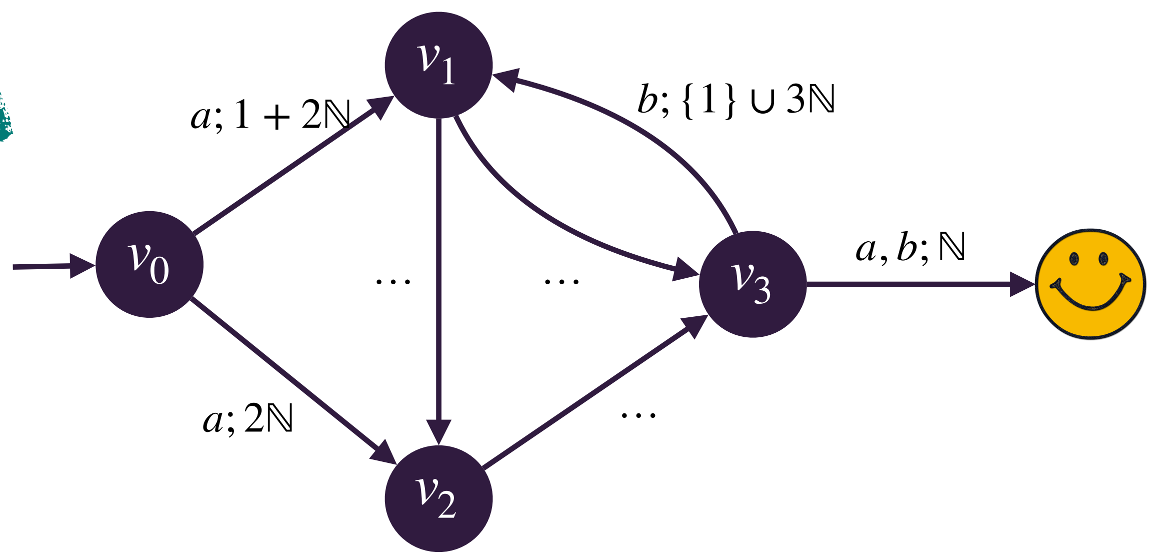
From languages to counting

φ objective



count

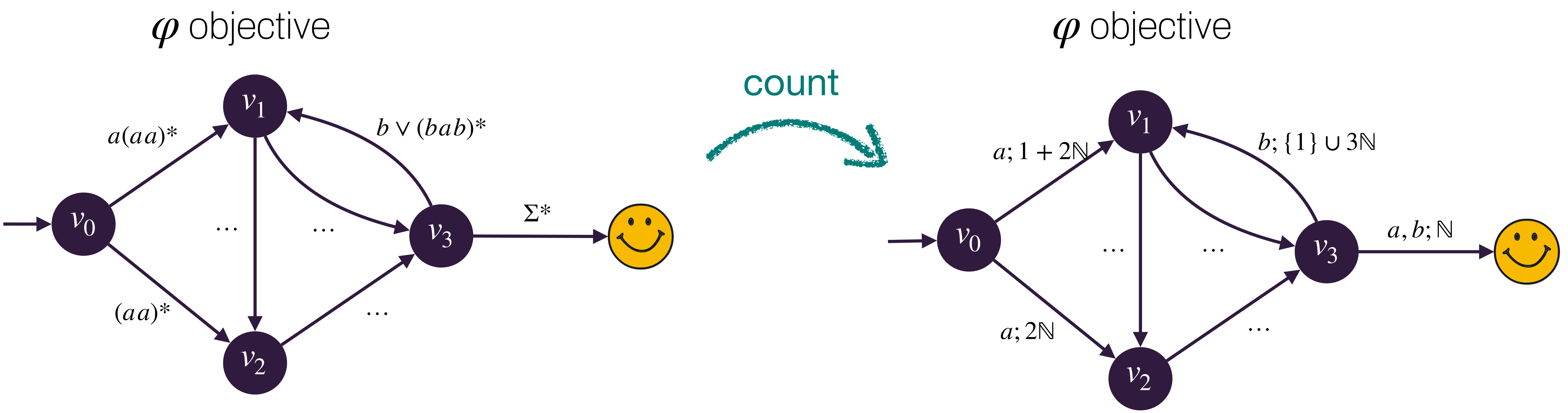
φ objective



How do we play this new game?

- The game starts at v_0
- The opponent chooses k (unknown to Gru)
- While (true)
  - At vertex v, Gru chooses an action and opponent chooses an edge $v \xrightarrow{a;S} v'$ with $k \in S$
  - The game proceeds from v'

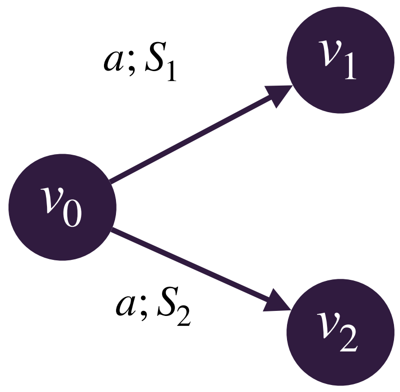
From languages to counting



Gru wins the language/original game iff he wins the counting game

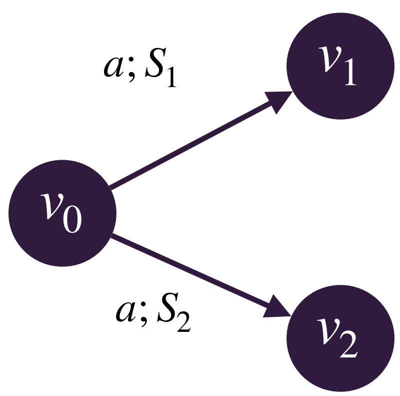
From counting to turn-based knowledge games

φ objective

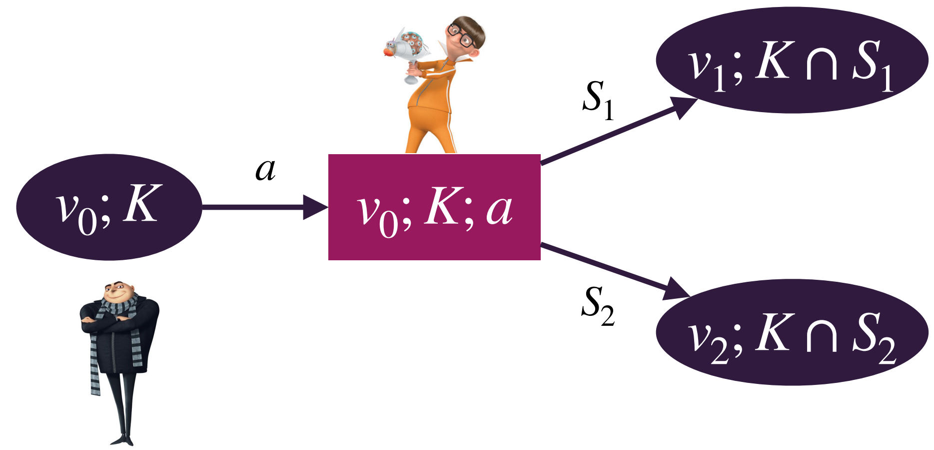


From counting to turn-based knowledge games

φ objective

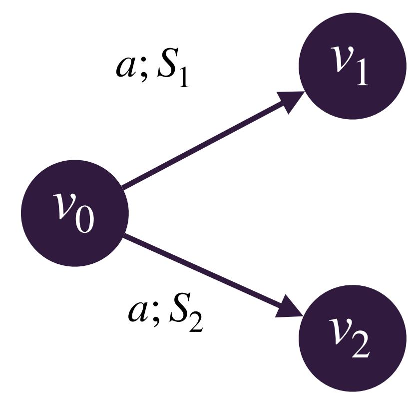


φ objective

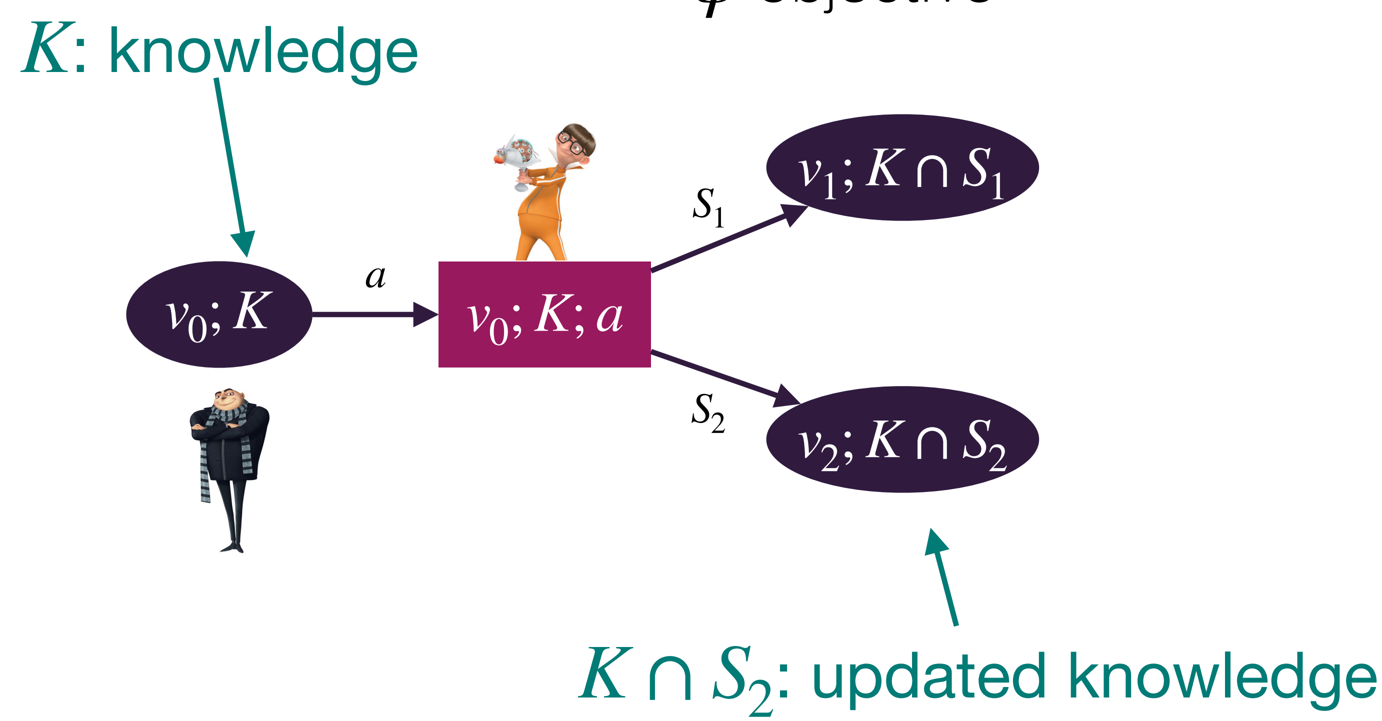


From counting to turn-based knowledge games

φ objective

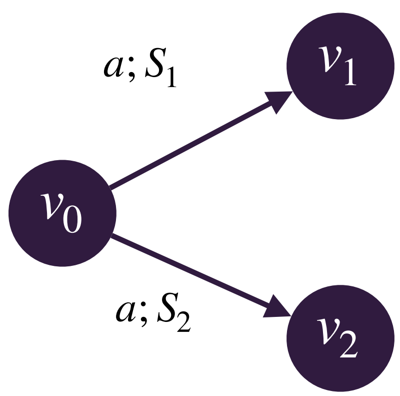


φ objective

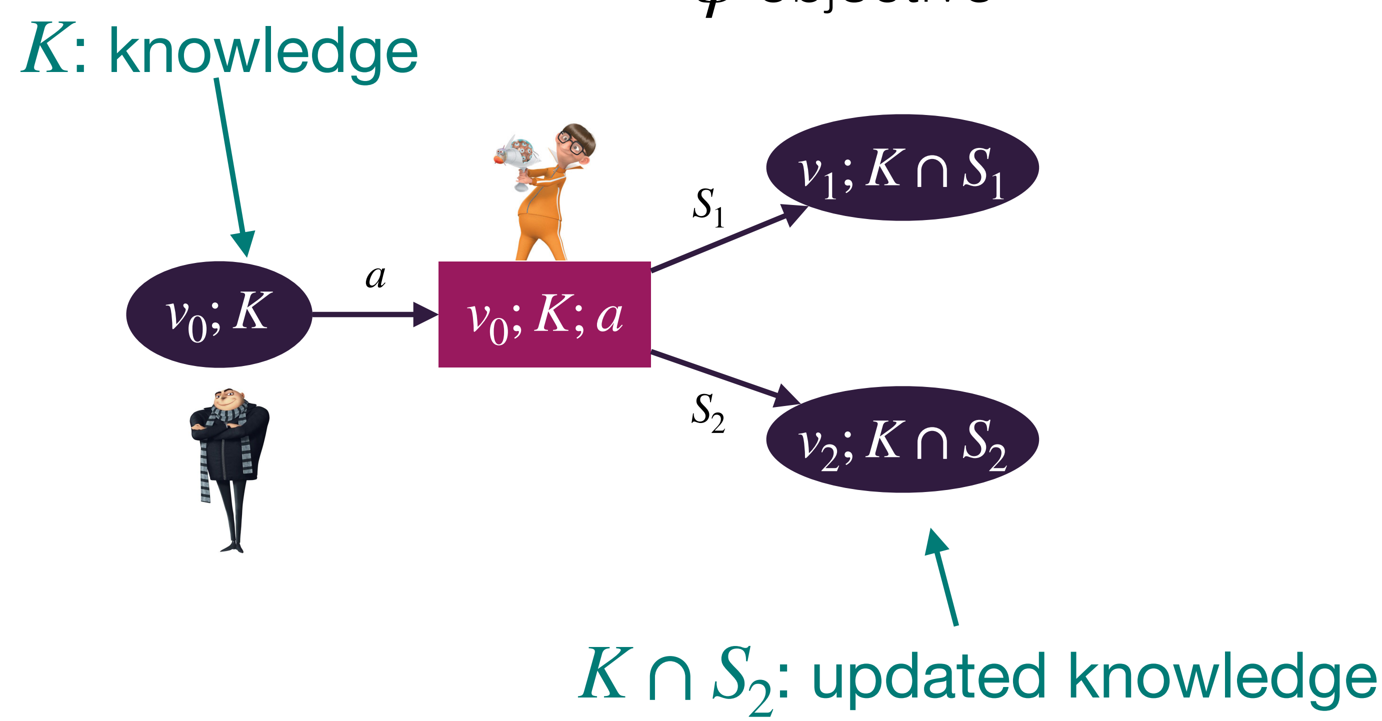


From counting to turn-based knowledge games

φ objective



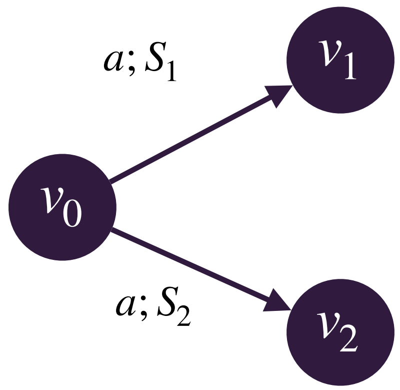
φ objective



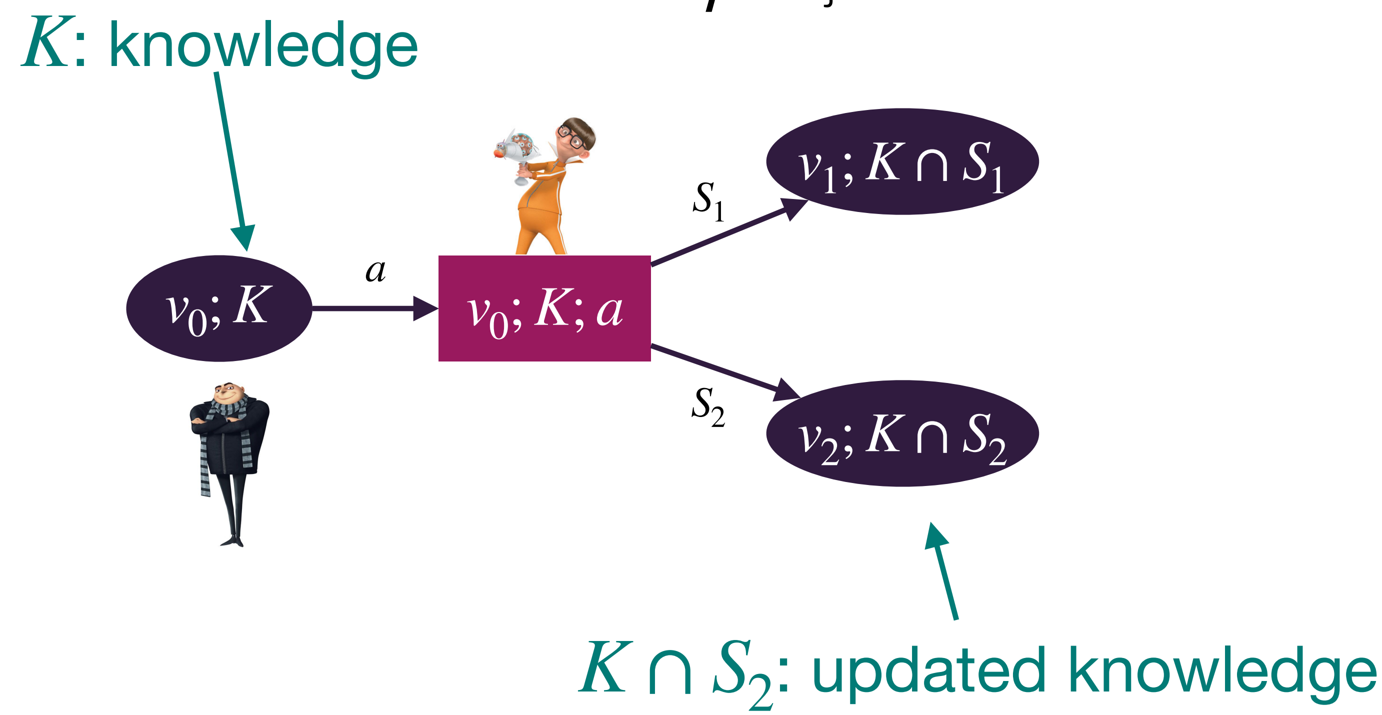
- Gru (Oval) chooses action
- Vector (Box) chooses semi linear set
- The game starts at $(v_0; \mathbb{N})$, and knowledge is updated at each round

From counting to turn-based knowledge games

φ objective



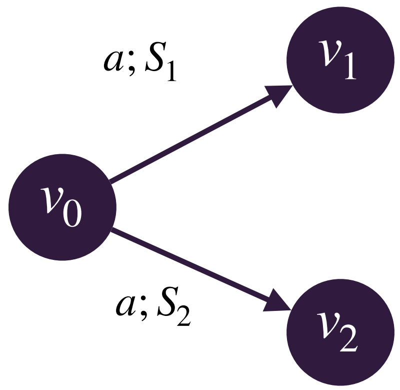
φ objective



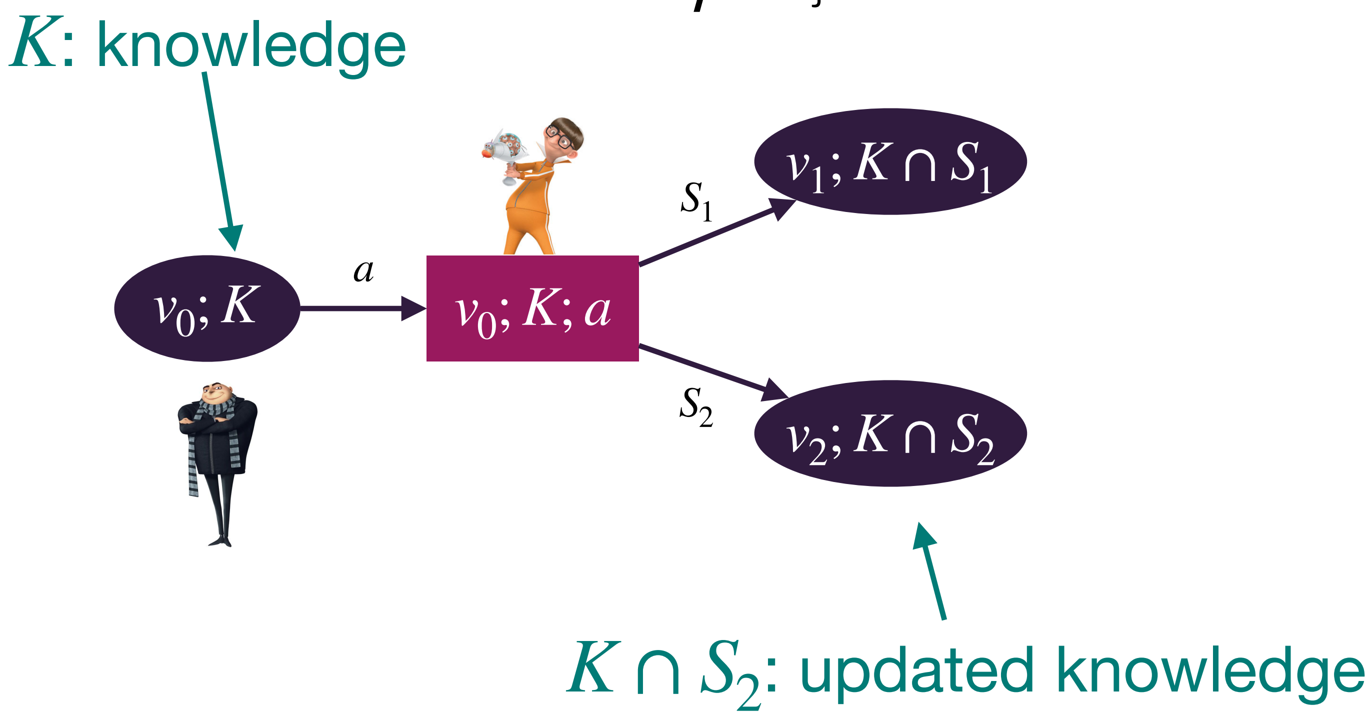
Gru wins the counting game iff he wins the knowledge game

From counting to turn-based knowledge games

φ objective



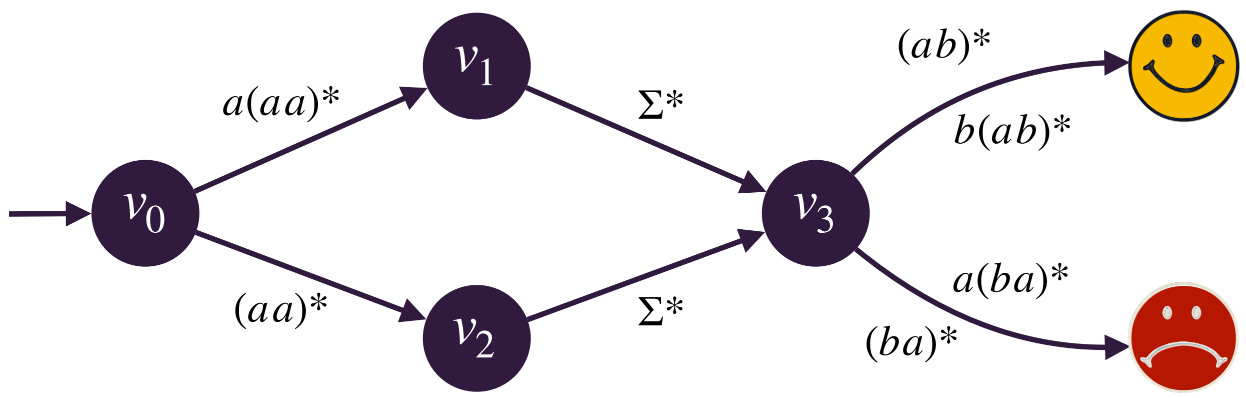
φ objective



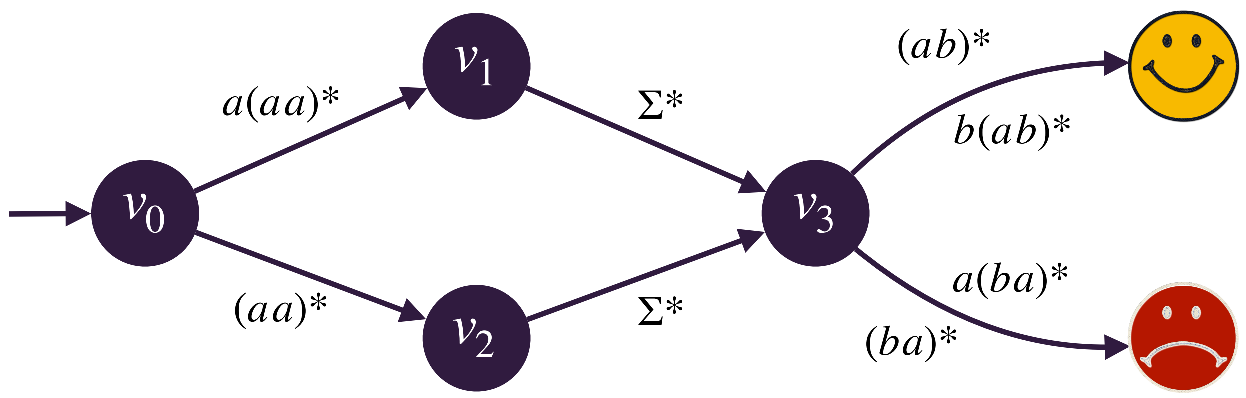
Gru wins the counting game iff he wins the knowledge game

Note : the complexity is that of solving turn-based knowledge games with objective φ
Example: polynomial-time w.r.t. its size for Reachability objectives

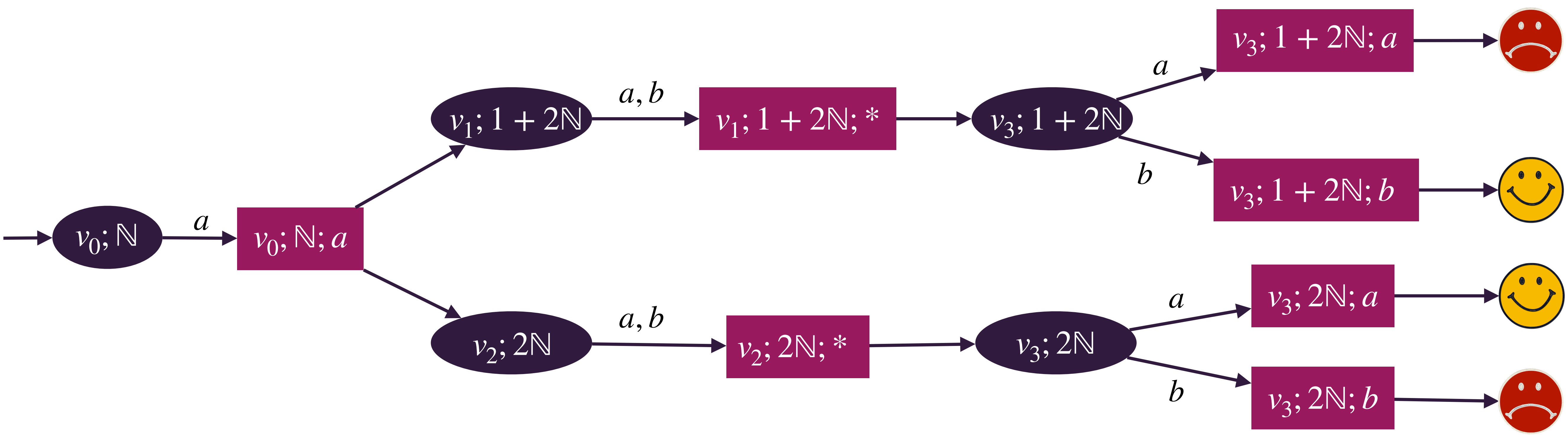
An example



An example

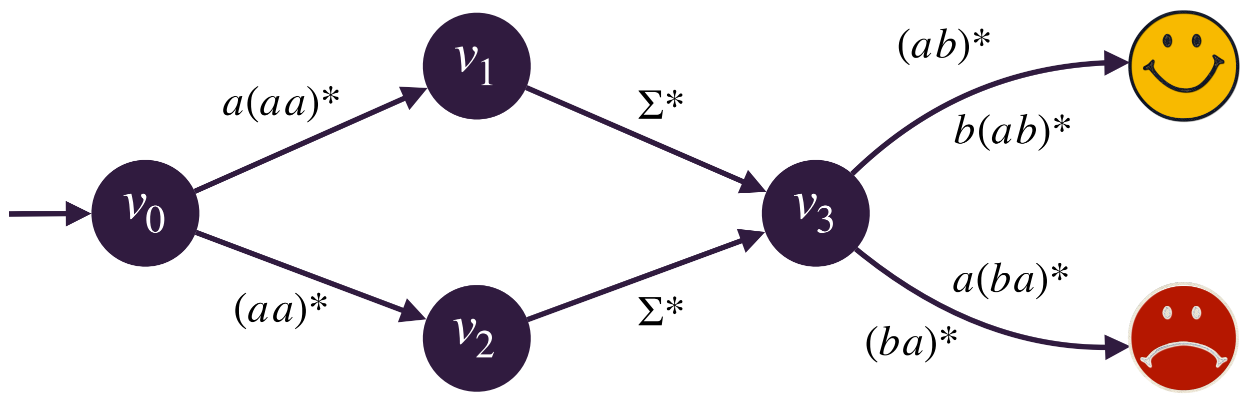


v_0
v_1
v_2
v_3
$a(aa)^*$
$(aa)^*$
Σ^*
Σ^*
$(ab)^*$
$b(ab)^*$
$a(ba)^*$
$(ba)^*$

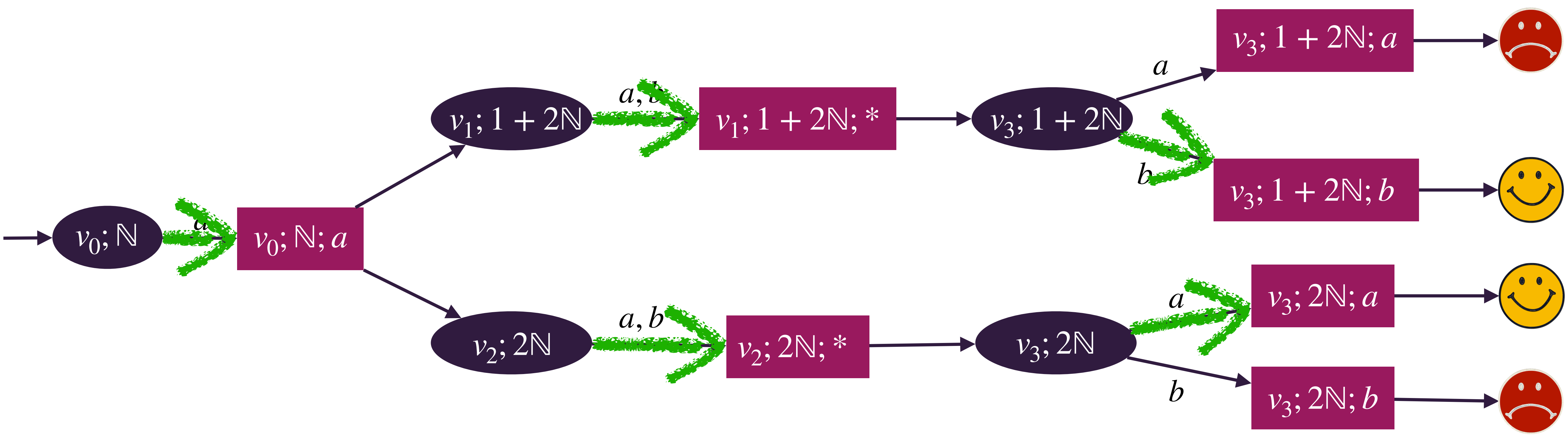


$v_0; \mathbb{N}$
a
$v_0; \mathbb{N}; a$
$v_1; 1 + 2\mathbb{N}$
a, b
$v_1; 1 + 2\mathbb{N}; *$
$v_3; 1 + 2\mathbb{N}$
a
$v_3; 1 + 2\mathbb{N}; a$
b
$v_3; 1 + 2\mathbb{N}; b$
$v_2; 2\mathbb{N}$
a, b
$v_2; 2\mathbb{N}; *$
$v_3; 2\mathbb{N}$
a
$v_3; 2\mathbb{N}; a$
b
$v_3; 2\mathbb{N}; b$

An example

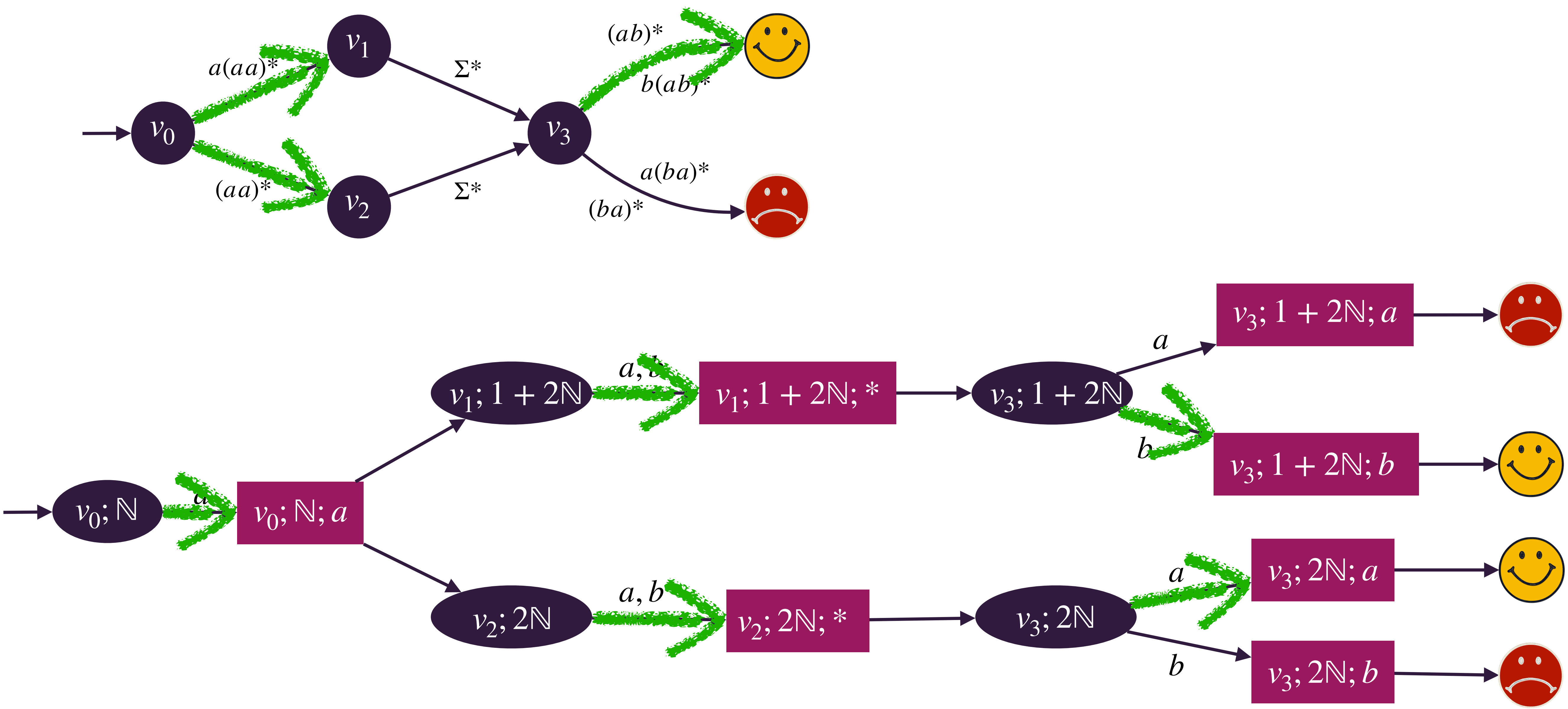


v_0
v_1
v_2
v_3
$a(aa)^*$
$(aa)^*$
Σ^*
Σ^*
$(ab)^*$
$b(ab)^*$
$a(ba)^*$
$(ba)^*$



$v_0; \mathbb{N}$
a
$v_0; \mathbb{N}; a$
$v_1; 1 + 2\mathbb{N}$
a, b
$v_1; 1 + 2\mathbb{N}; *$
$v_3; 1 + 2\mathbb{N}$
a
$v_3; 1 + 2\mathbb{N}; a$
b
$v_3; 1 + 2\mathbb{N}; b$
$v_2; 2\mathbb{N}$
a, b
$v_2; 2\mathbb{N}; *$
$v_3; 2\mathbb{N}$
a
$v_3; 2\mathbb{N}; a$
b
$v_3; 2\mathbb{N}; b$

An example



v_1
v_0
v_2
v_3
$a(aa)^*$
$(aa)^*$
Σ^*
Σ^*
$(ab)^*$
$b(ab)^*$
$a(ba)^*$
$(ba)^*$
$v_0; \mathbb{N}$
a
$v_0; \mathbb{N}; a$
$v_1; 1+2\mathbb{N}$
a, b
$v_1; 1+2\mathbb{N}; *$
$v_3; 1+2\mathbb{N}$
a
$v_3; 1+2\mathbb{N}; a$
b
$v_3; 1+2\mathbb{N}; b$
$v_2; 2\mathbb{N}$
a, b
$v_2; 2\mathbb{N}; *$
$v_3; 2\mathbb{N}$
a
$v_3; 2\mathbb{N}; a$
b
$v_3; 2\mathbb{N}; b$

The results

Complexity results

The crowd controller problem is decidable and has the following complexity for Reachability objectives:

<table>
<tr><th></th><th>Deterministic arenas</th><th>Non-deterministic arenas</th></tr>
<tr><td>Intervals</td><td colspan="2">PTIME-complete</td></tr>
<tr><td>Finite unions of intervals</td><td>NP-complete</td><td>PSPACE-complete</td></tr>
<tr><td>Semilinear sets</td><td colspan="2" rowspan="2">PSPACE-complete</td></tr>
<tr><td>Reg/CF languages</td></tr>
</table>

The results

Complexity results

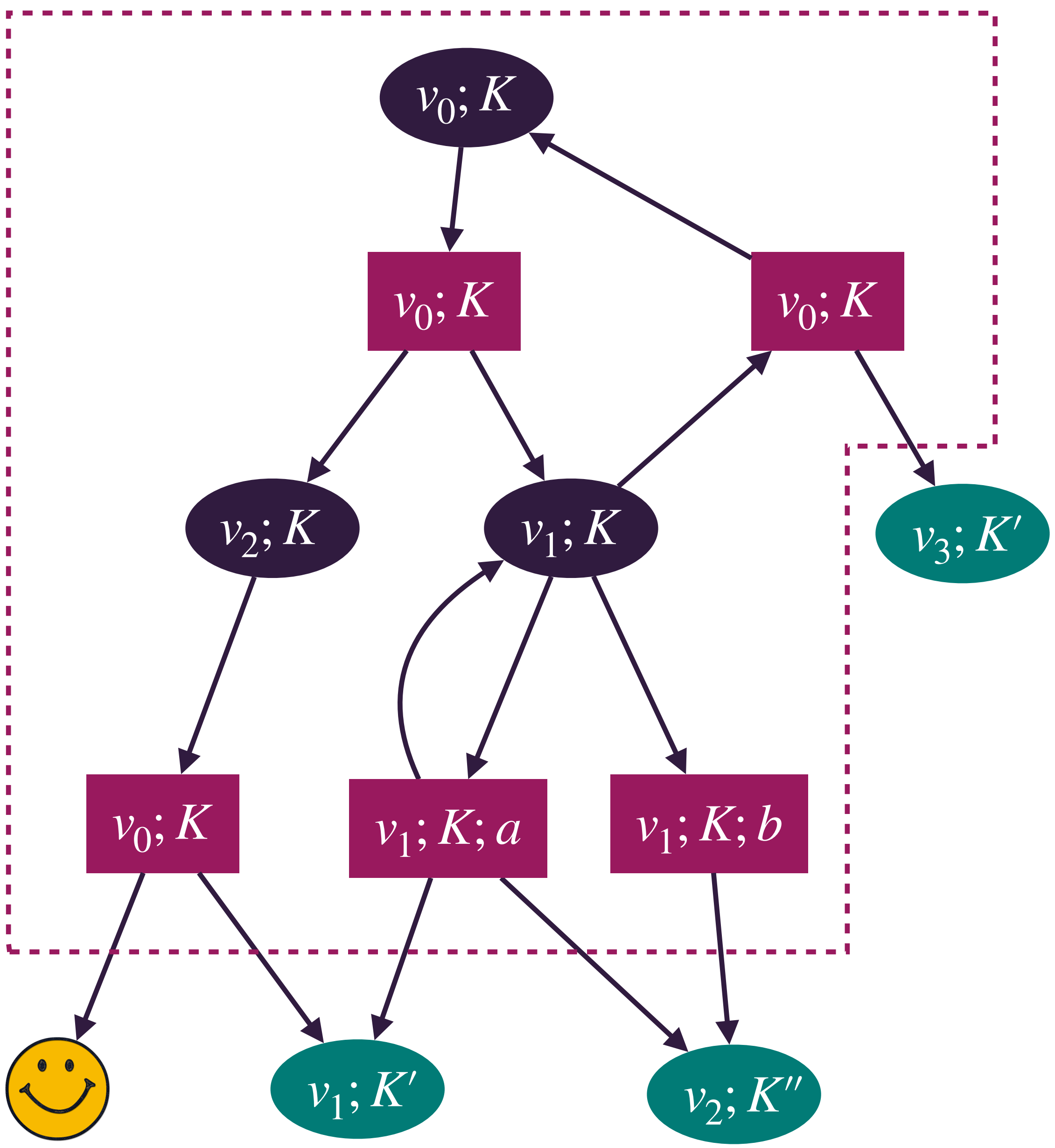
The crowd controller problem is decidable and has the following complexity for Reachability objectives:

	Deterministic arenas	**Non-deterministic arenas**
Intervals	PTIME-complete	
Finite unions of intervals	NP-complete	PSPACE-complete
Semilinear sets	PSPACE-complete	
Reg/CF languages		

- Each knowledge is an intersection of (atomic) constraints used in the game
- The number of possible knowledges is therefore at most exponential in the number of (atomic) constraints used in the game
  - Semilinear sets: the knowledge game is at most exponential in the number of semilinear sets
- Finite unions of intervals: the knowledge game is at most exponential in the number of endpoints
- Intervals: the knowledge game is quadratic in the number of endpoints of the intervals

PSPACE algorithm - 1

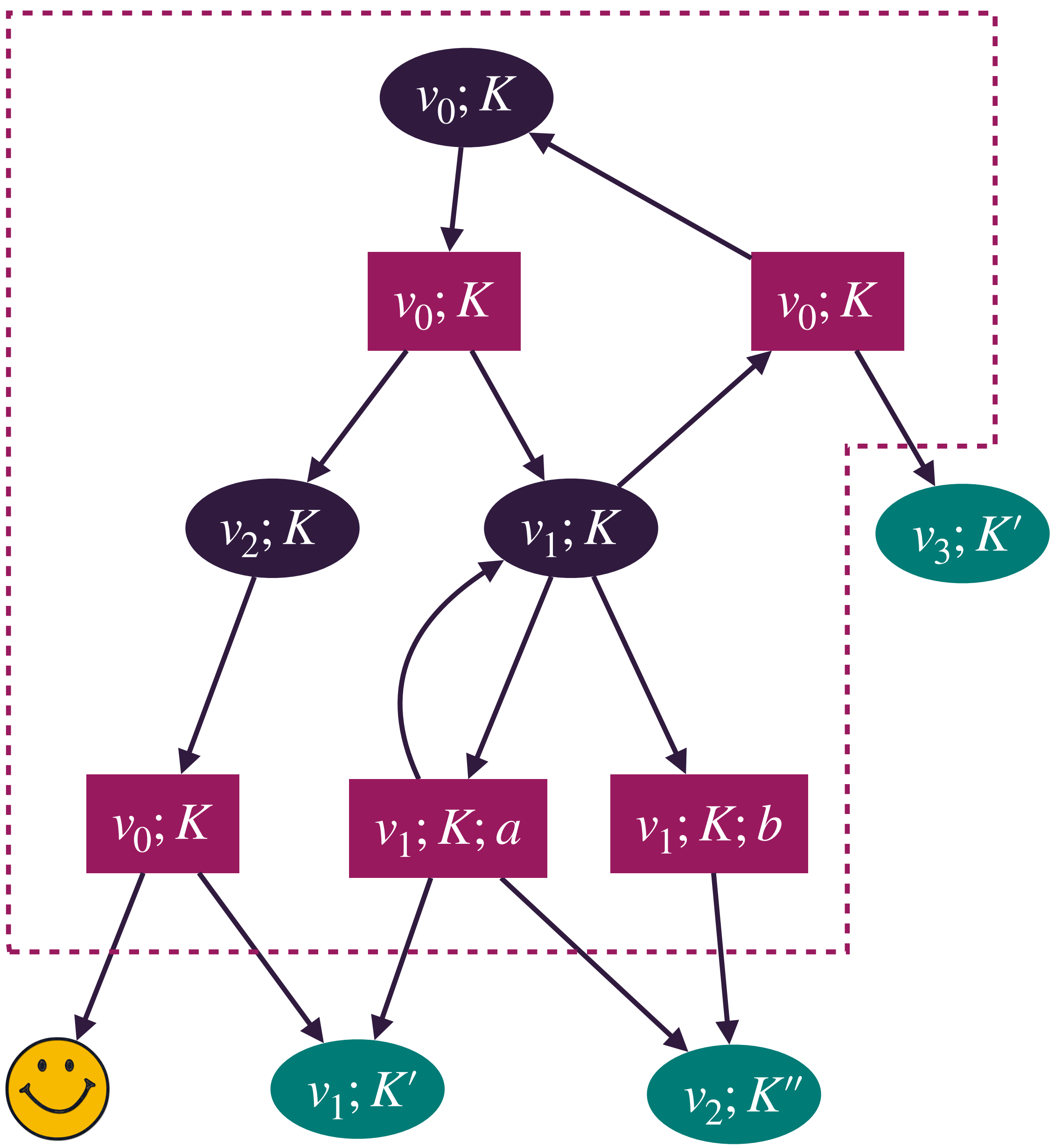
Subgame associated with knowledge K



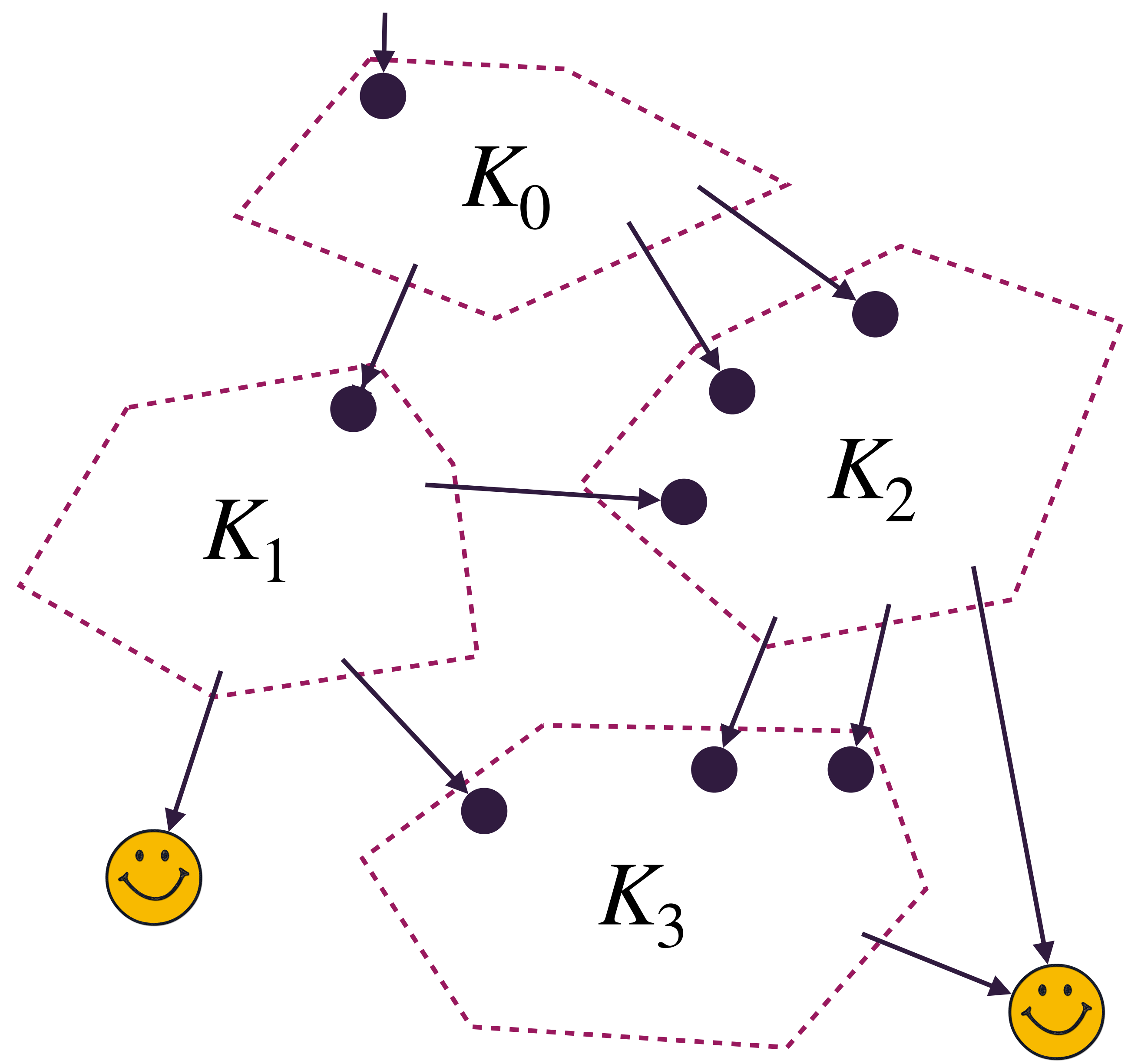
$K', K'' \subsetneq K$

PSPACE algorithm - 1

Subgame associated with knowledge K

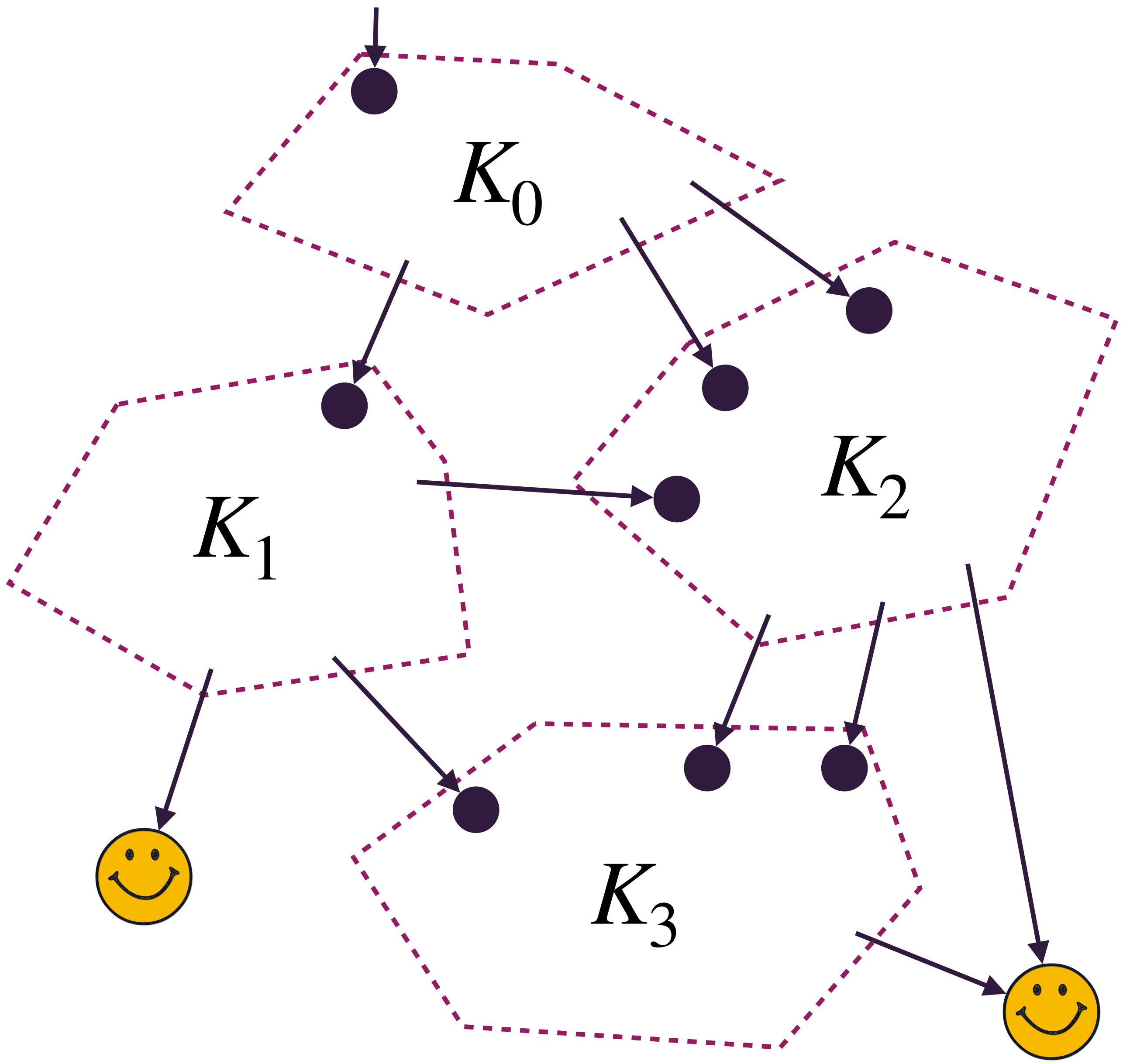


$K', K'' \subsetneq K$

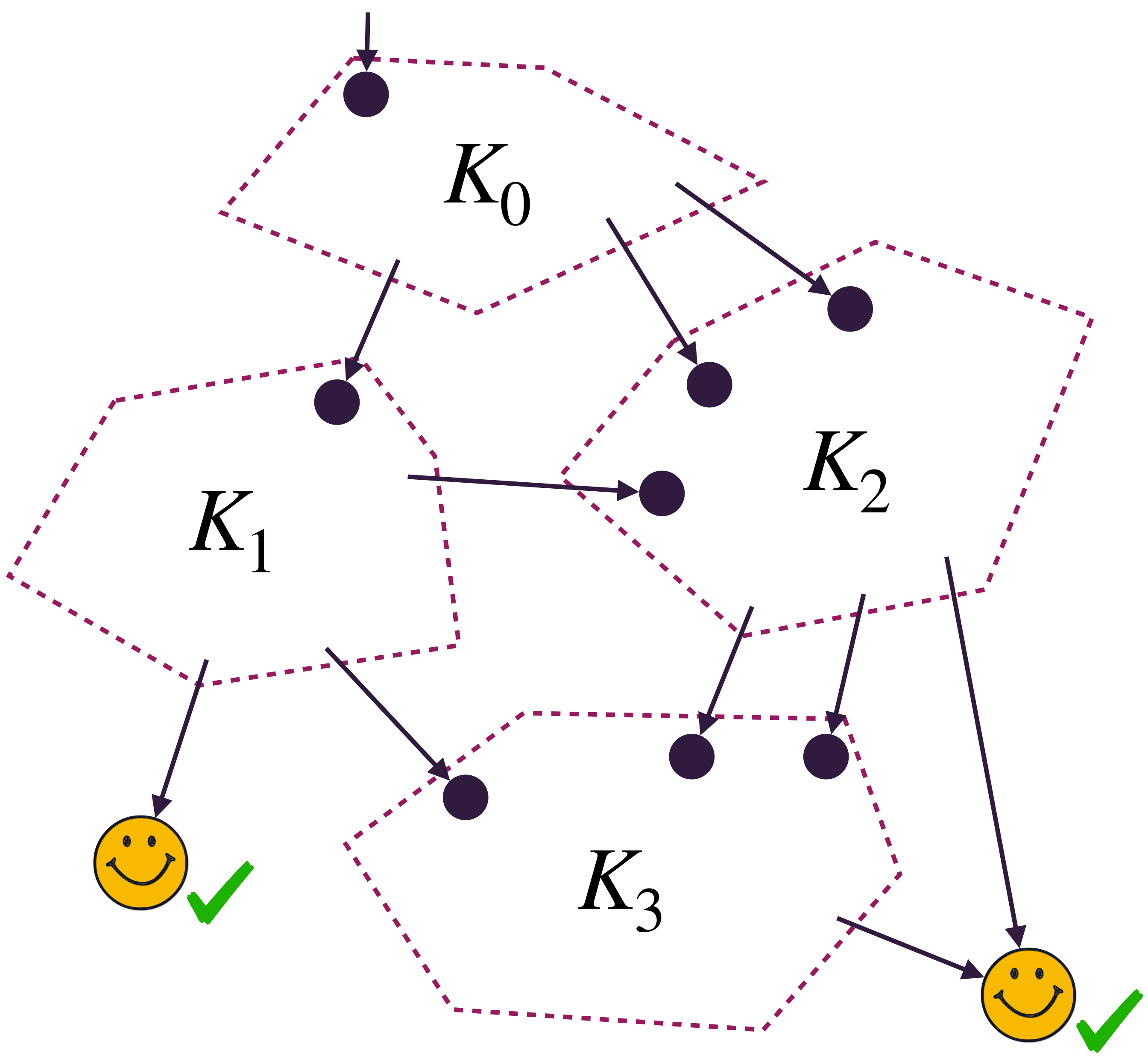


$K_3 \subsetneq K_1, K_2 \subsetneq K_0$

PSPACE algorithm - 2

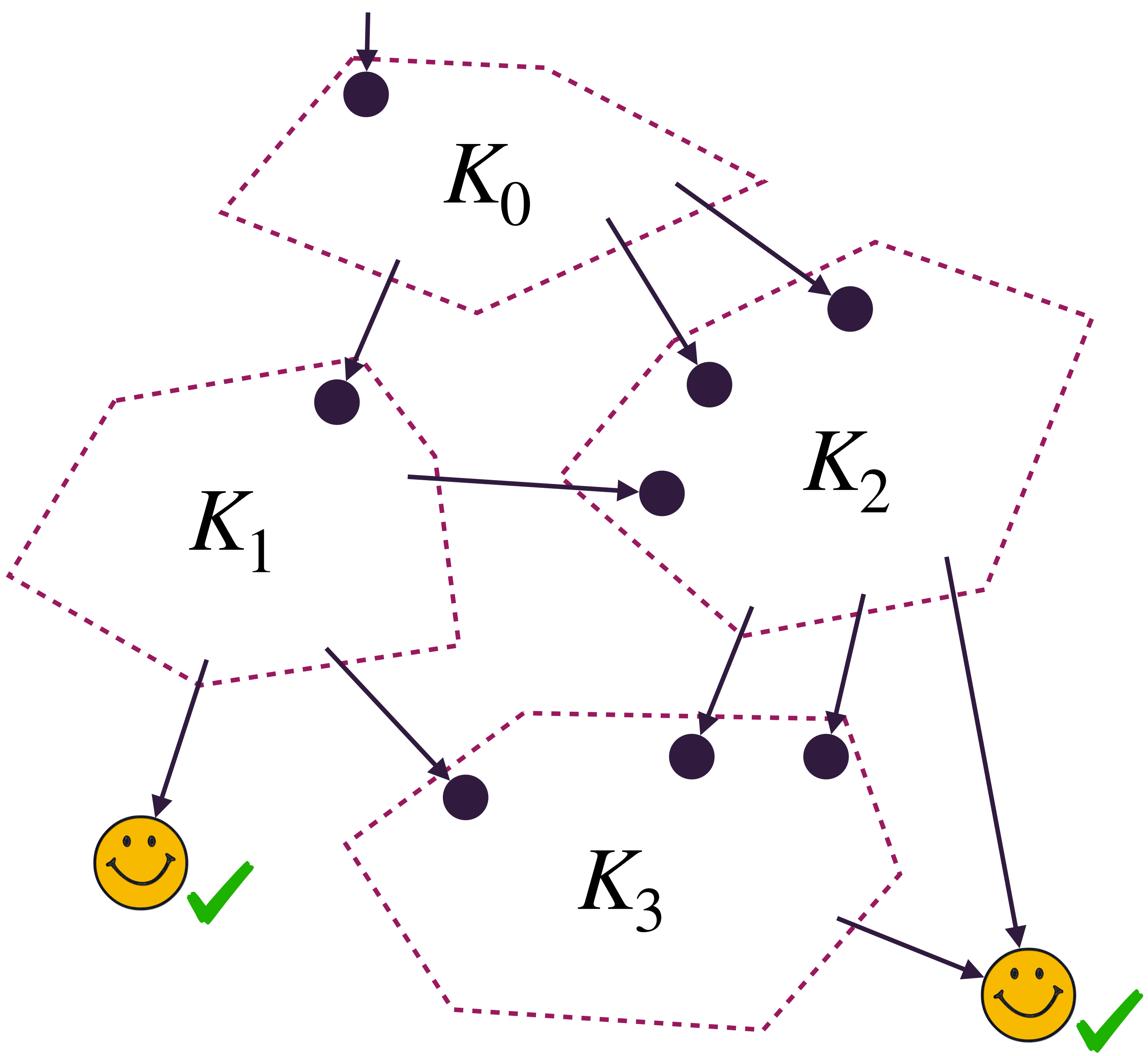


PSPACE algorithm - 2



Bottom-up tag of winning states

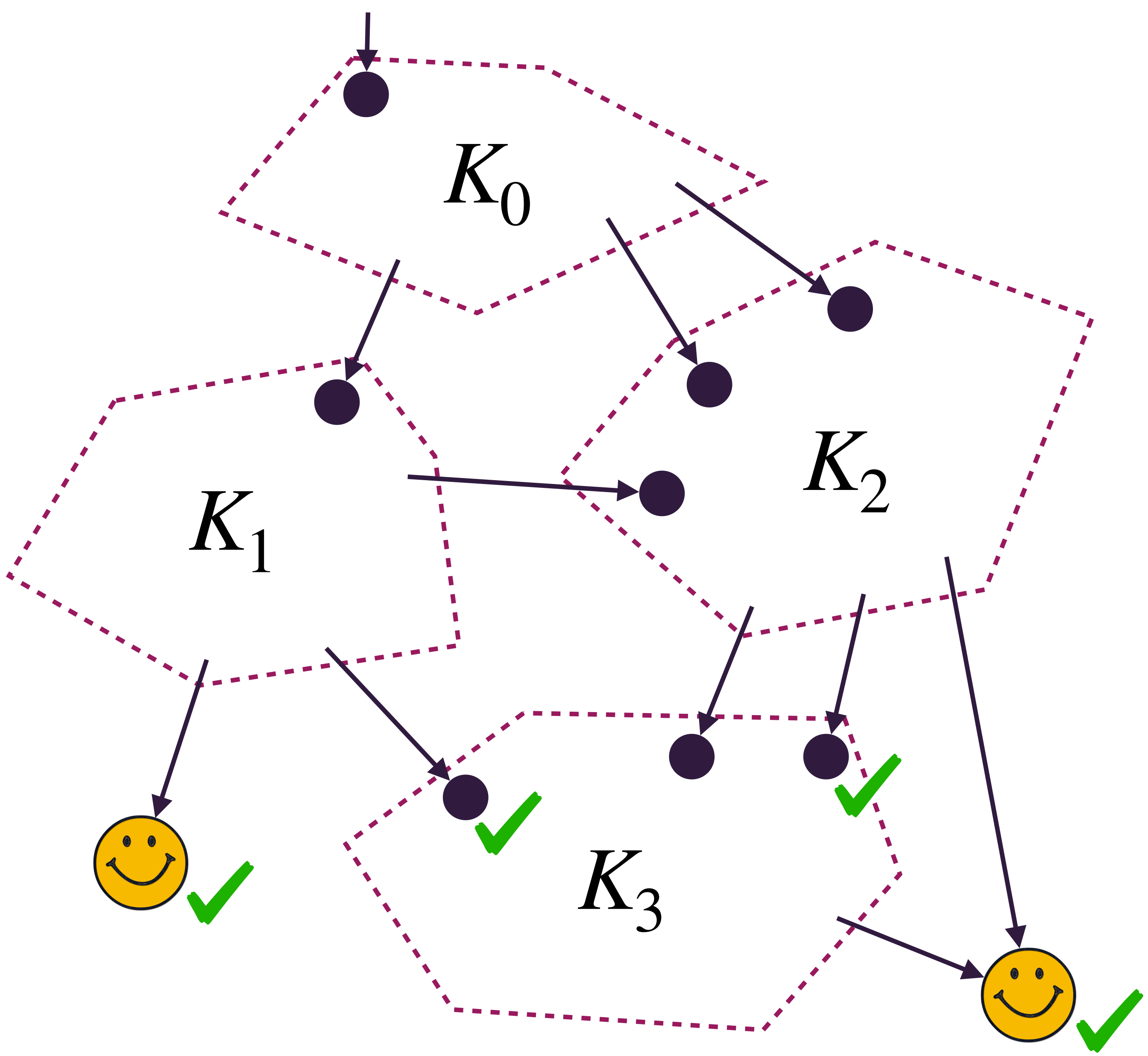
PSPACE algorithm - 2



Bottom-up tag of winning states

- Start at subgame with knowledge K_3:
  - Objective: φ

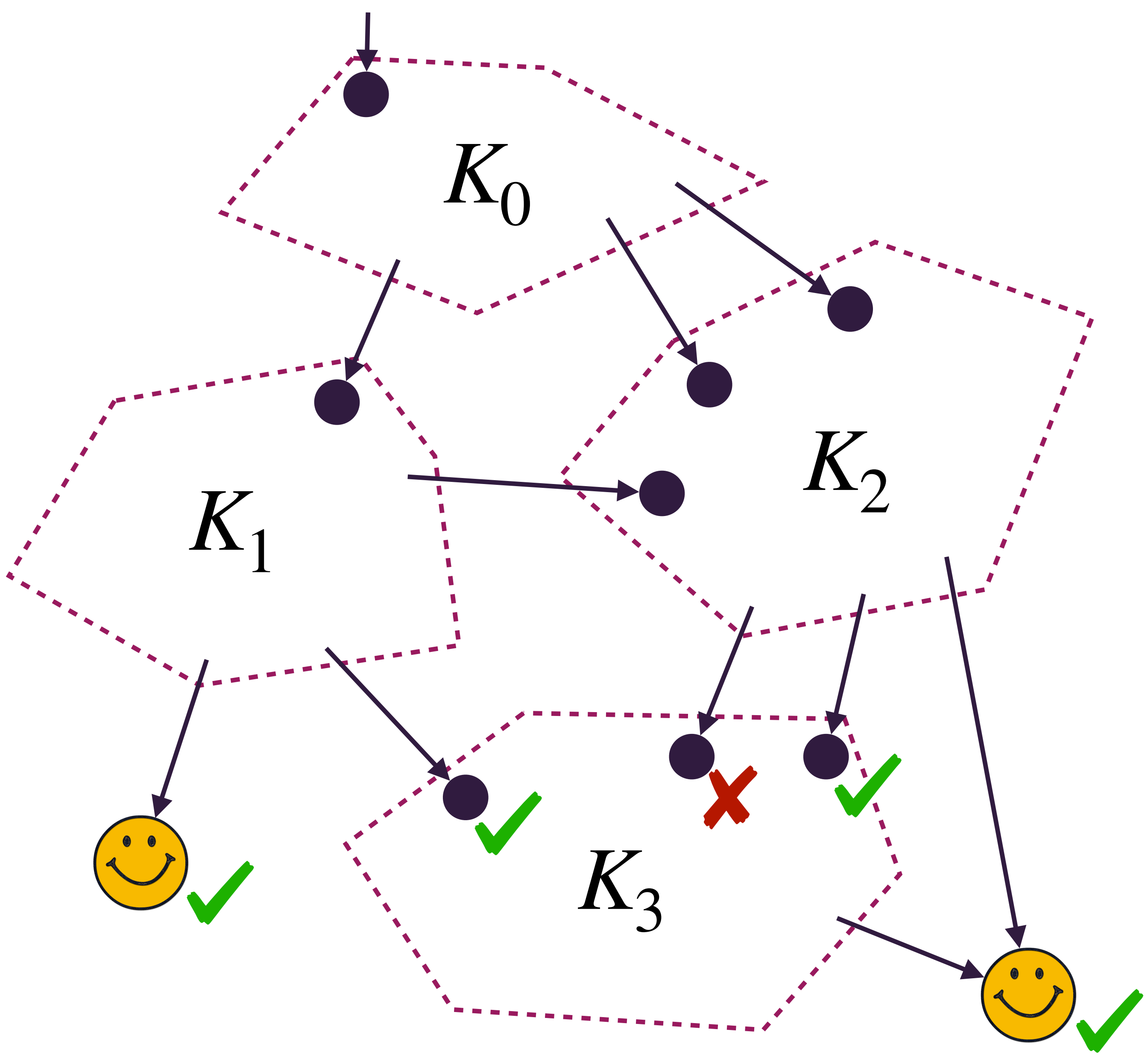
PSPACE algorithm - 2



Bottom-up tag of winning states

- Start at subgame with knowledge K_3:
  - Objective: φ
  - Tag winning states with ✓

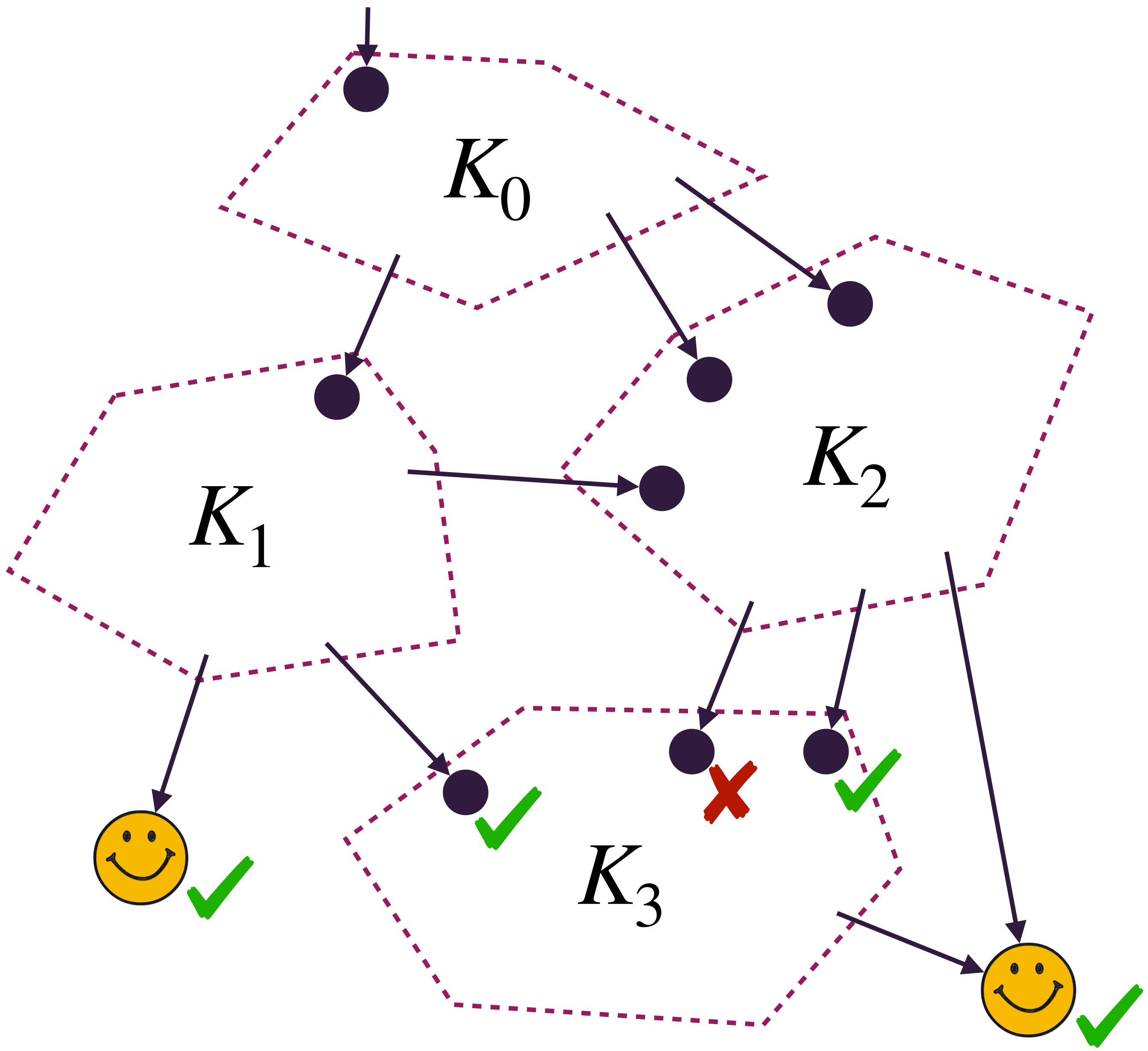
PSPACE algorithm - 2



Bottom-up tag of winning states

- Start at subgame with knowledge K_3:
  - Objective: φ
  - Tag winning states with ✓
  - Tag losing states with ✗

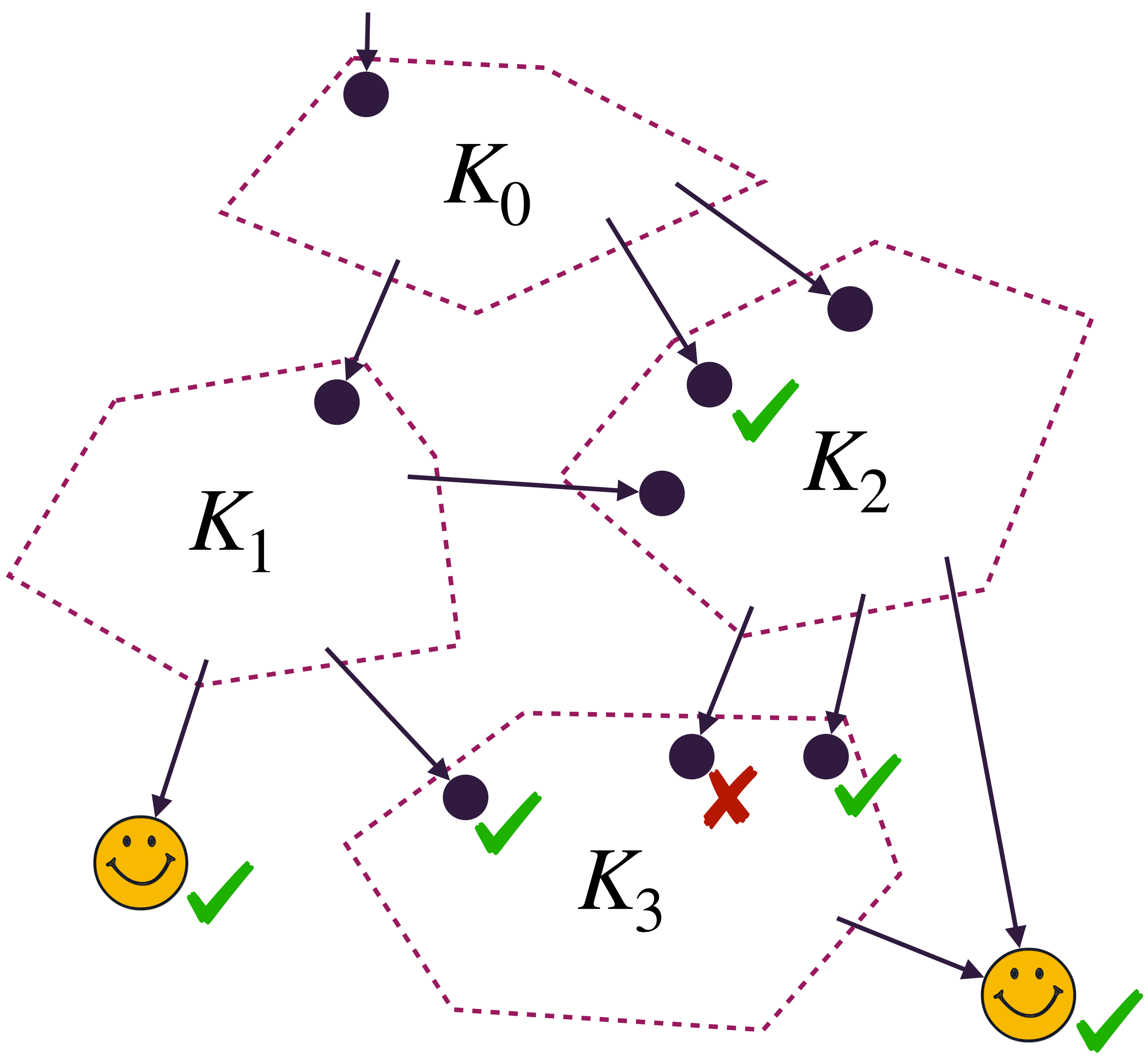
PSPACE algorithm - 2



Bottom-up tag of winning states

- Start at subgame with knowledge K_3:
  - Objective: φ
  - Tag winning states with ✓
  - Tag losing states with ✗
- Go to subgame with knowledge K_2:
  - Objective: φ or Reach(✓)

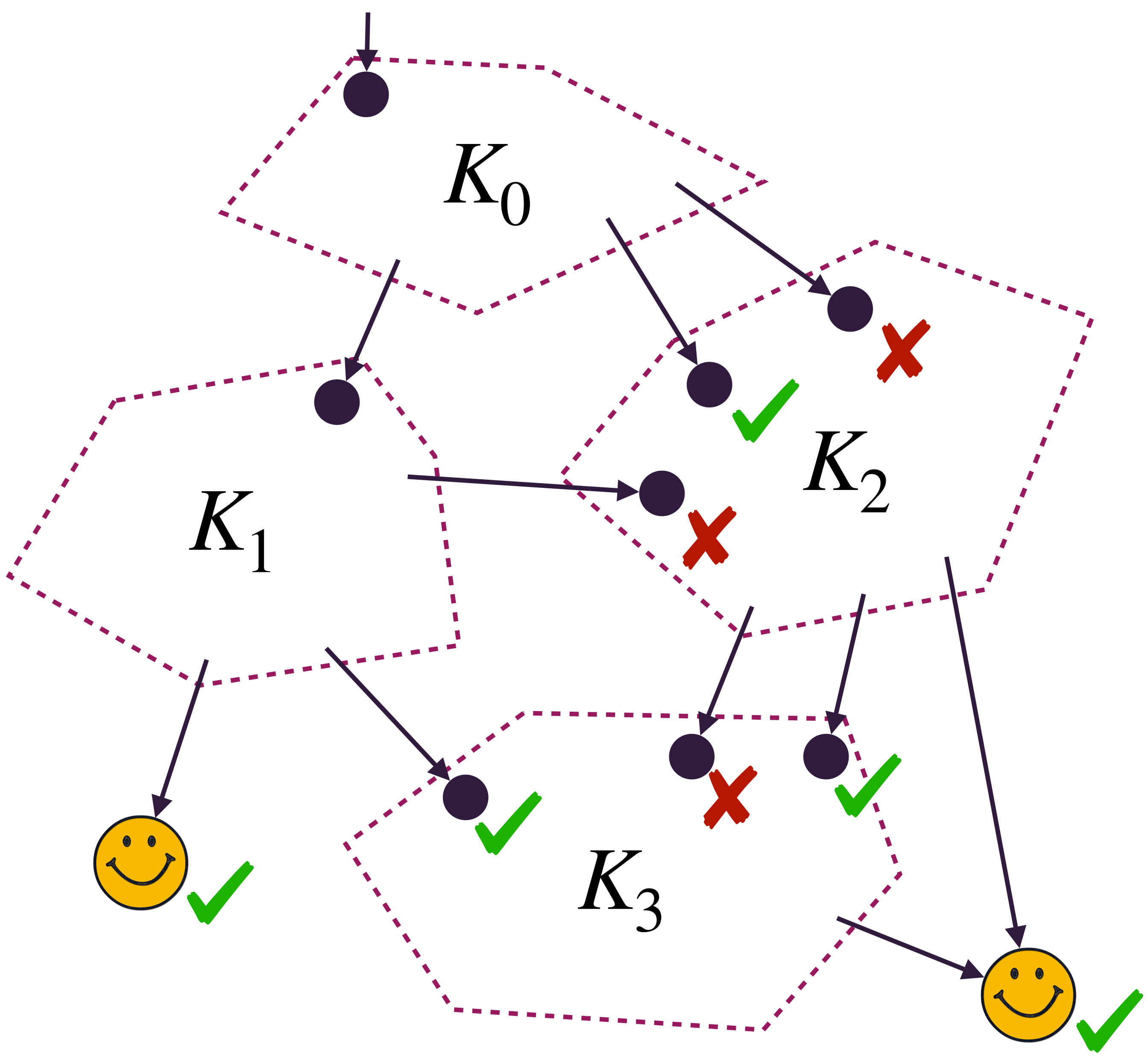
PSPACE algorithm - 2



Bottom-up tag of winning states

- Start at subgame with knowledge K_3:
  - Objective: φ
  - Tag winning states with ✓
  - Tag losing states with ✗
- Go to subgame with knowledge K_2:
  - Objective: φ or Reach(✓)
  - Tag winning states with ✓

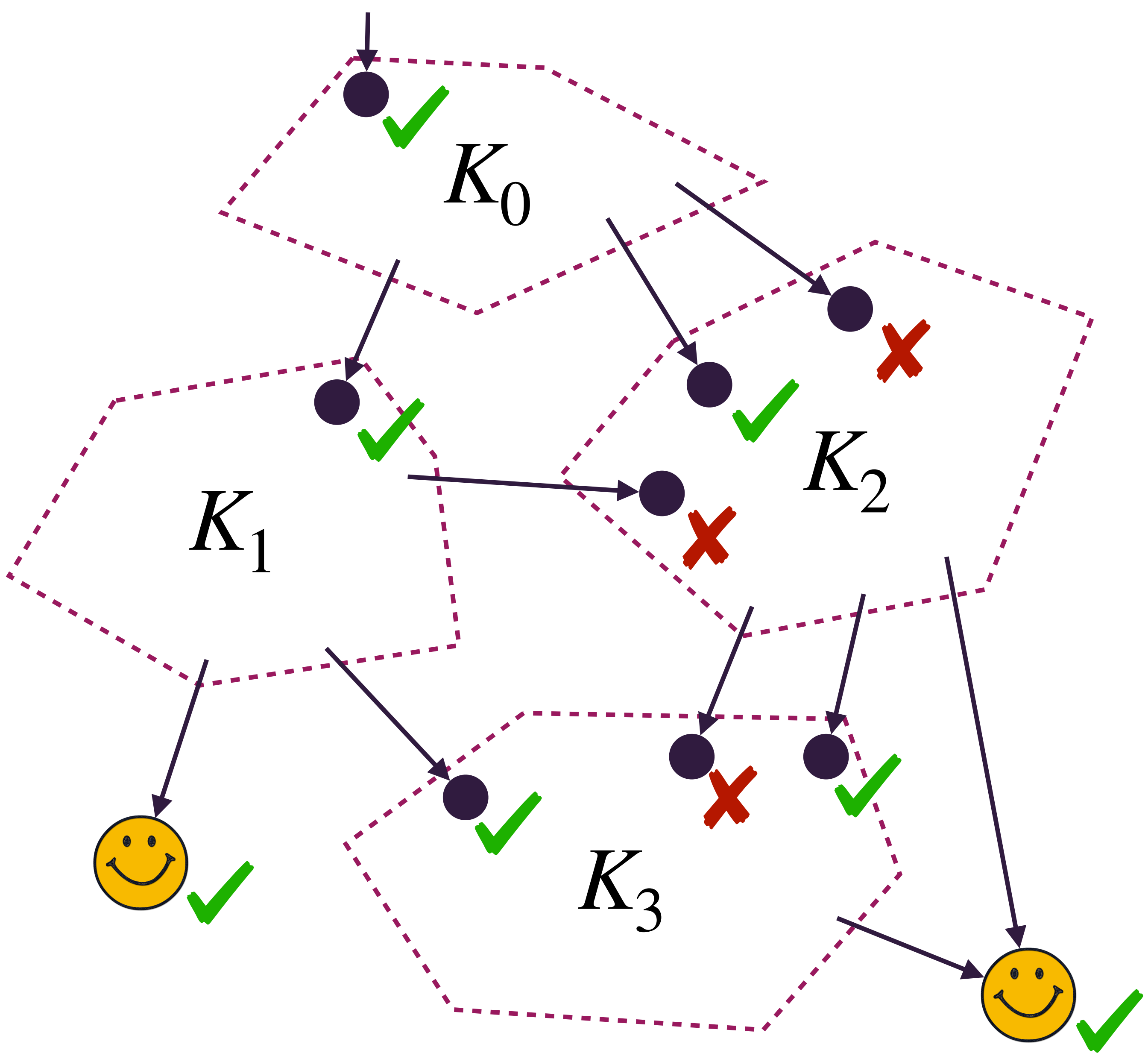
PSPACE algorithm - 2



Bottom-up tag of winning states

- Start at subgame with knowledge K_3:
  - Objective: φ
  - Tag winning states with ✓
  - Tag losing states with ✗
- Go to subgame with knowledge K_2:
  - Objective: φ or Reach(✓)
  - Tag winning states with ✓
  - Tag losing states with ✗

PSPACE algorithm - 2



Bottom-up tag of winning states

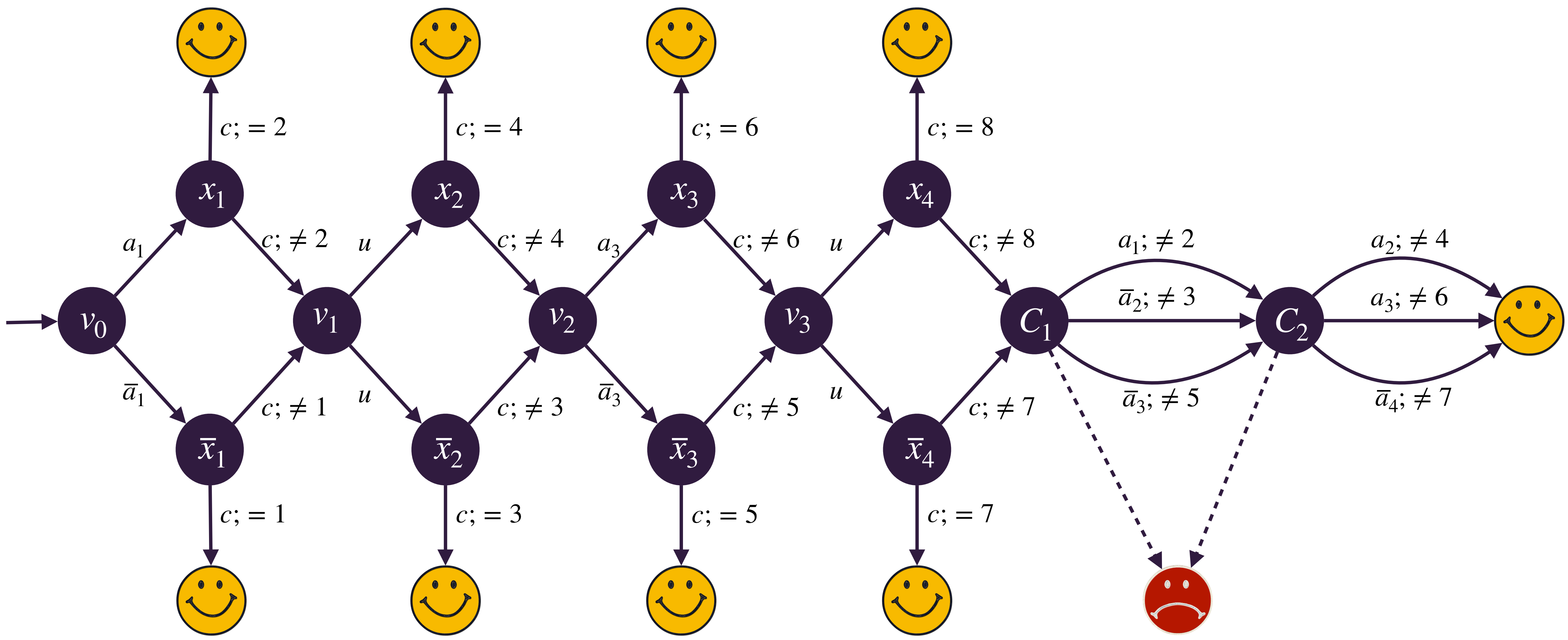
- Start at subgame with knowledge K_3:
  - Objective: φ
  - Tag winning states with ✓
  - Tag losing states with ✗
- Go to subgame with knowledge K_2:
  - Objective: φ or Reach(✓)
  - Tag winning states with ✓
  - Tag losing states with ✗
- Etc...

PSPACE-hardness - 1

QSAT formula $\psi = \exists x_1 \forall x_2 \exists x_3 \forall x_4 \cdot (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_2 \vee x_3 \vee \neg x_4)$

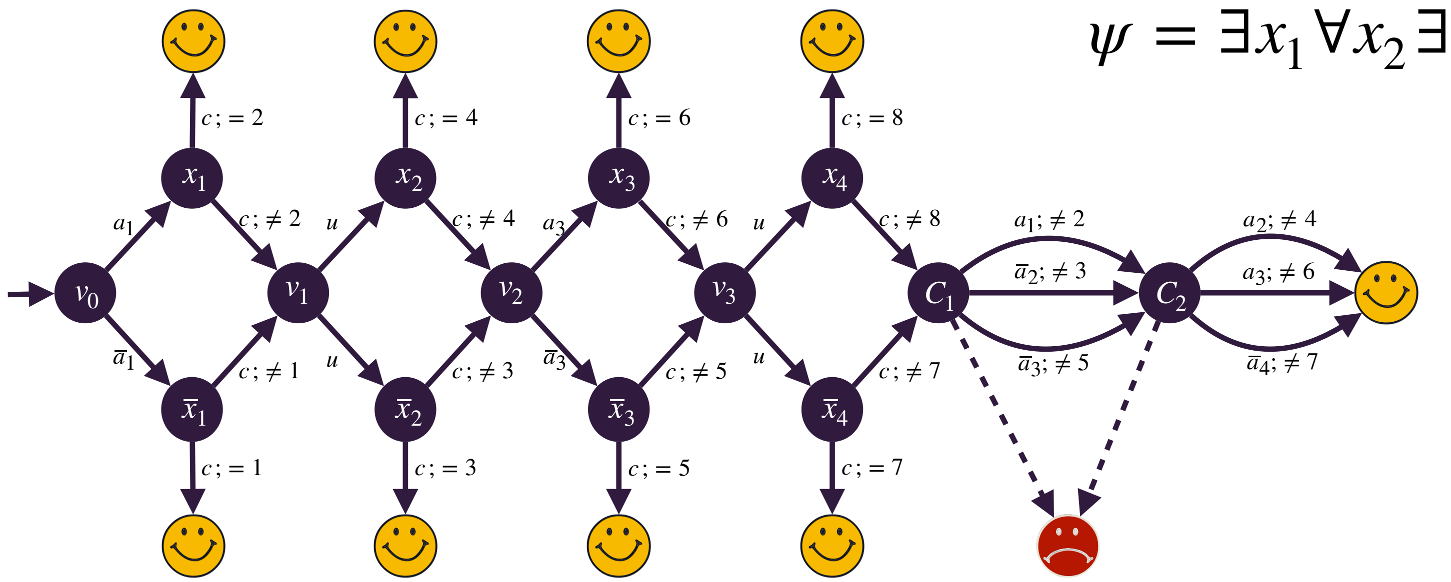
PSPACE-hardness - 1

QSAT formula $\psi = \exists x_1 \forall x_2 \exists x_3 \forall x_4 \cdot (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_2 \vee x_3 \vee \neg x_4)$



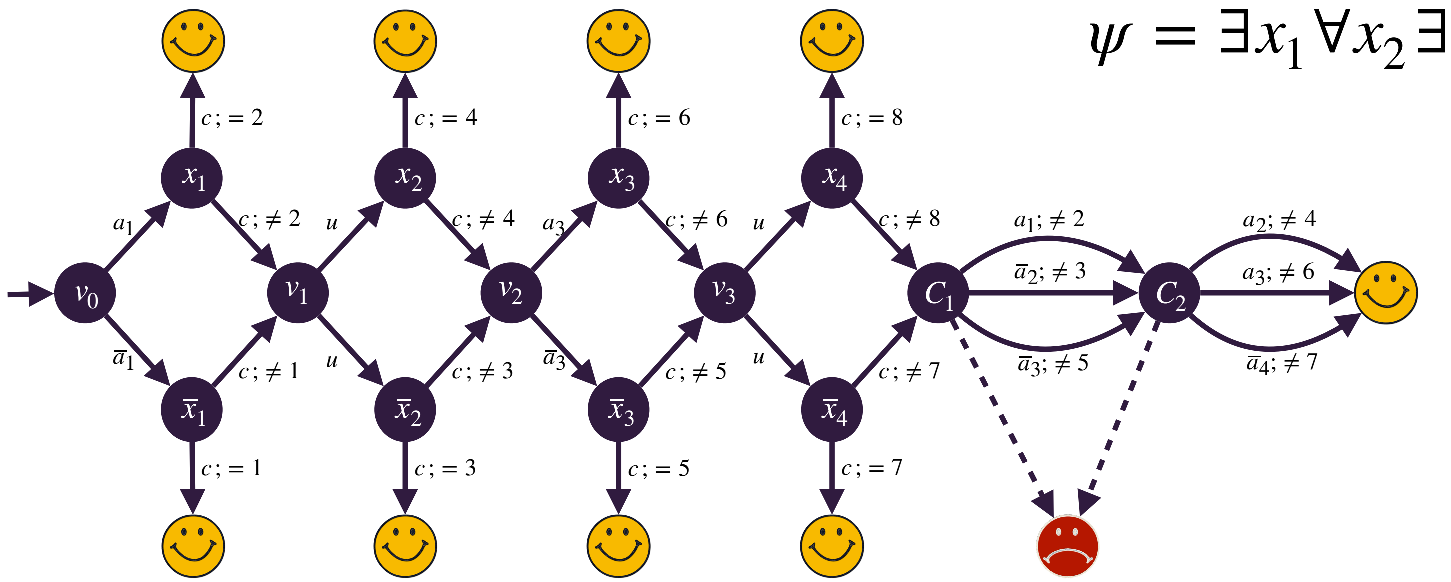
PSPACE-hardness - 2

$$\psi = \exists x_1 \forall x_2 \exists x_3 \forall x_4 \cdot (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_2 \vee x_3 \vee \neg x_4)$$



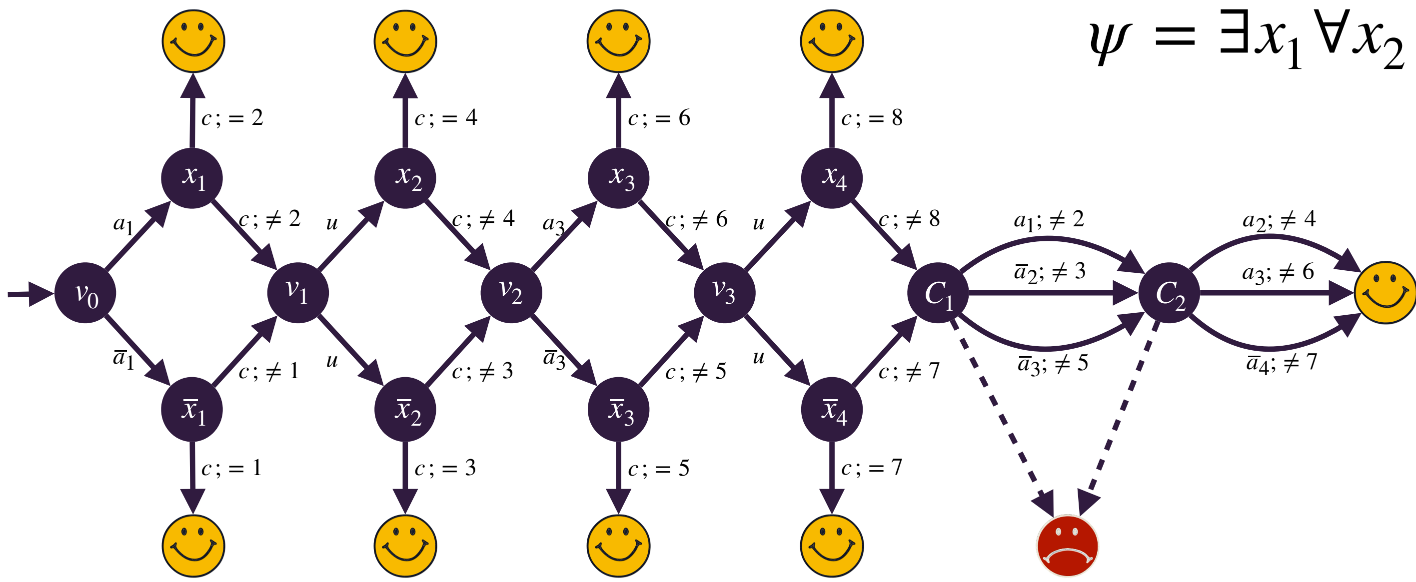
PSPACE-hardness - 2

$$\psi = \exists x_1 \forall x_2 \exists x_3 \forall x_4 \cdot (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_2 \vee x_3 \vee \neg x_4)$$



Strategy for Gru if ψ is true

PSPACE-hardness - 2

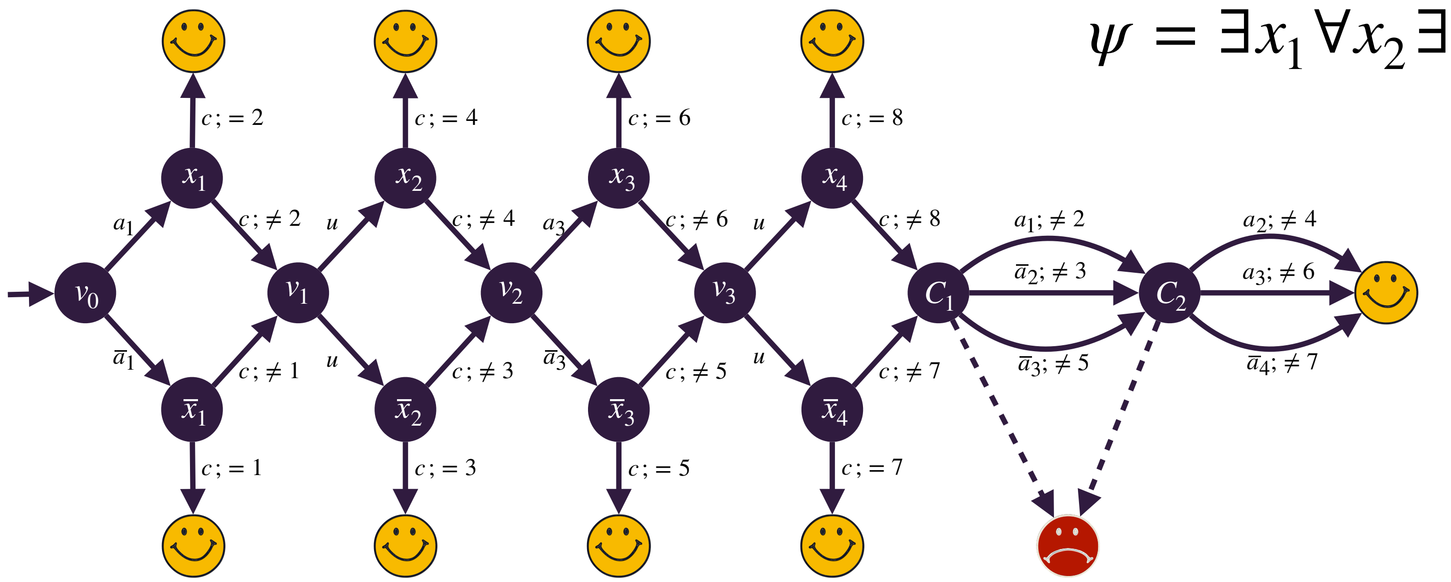


$$\psi = \exists x_1 \forall x_2 \exists x_3 \forall x_4 \cdot (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_2 \vee x_3 \vee \neg x_4)$$

Strategy for Gru if ψ is true

- At v_0, play the correct assignment, say false (i.e. $\overline{a}_1$), reaching $\overline{x}_1$

PSPACE-hardness - 2

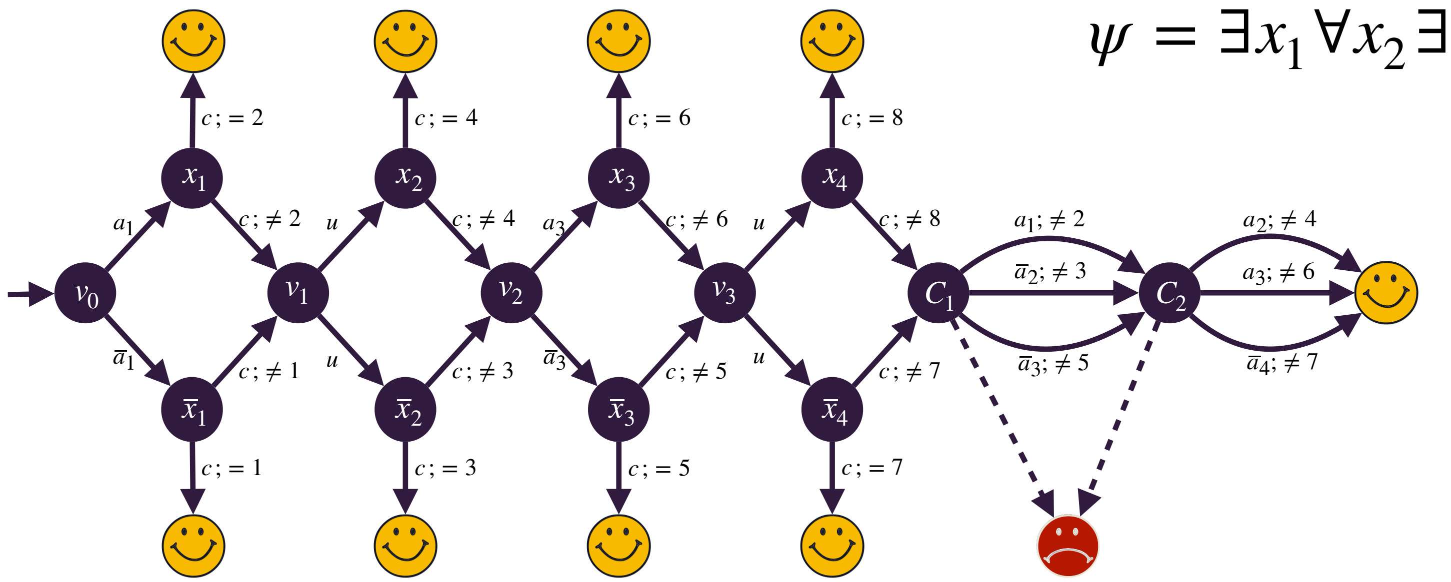


$$\psi = \exists x_1 \forall x_2 \exists x_3 \forall x_4 \cdot (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_2 \vee x_3 \vee \neg x_4)$$

Strategy for Gru if ψ is true

- At v_0, play the correct assignment, say false (i.e. $\overline{a}_1$), reaching $\overline{x}_1$
  - If $k = 1$, then go to 

PSPACE-hardness - 2

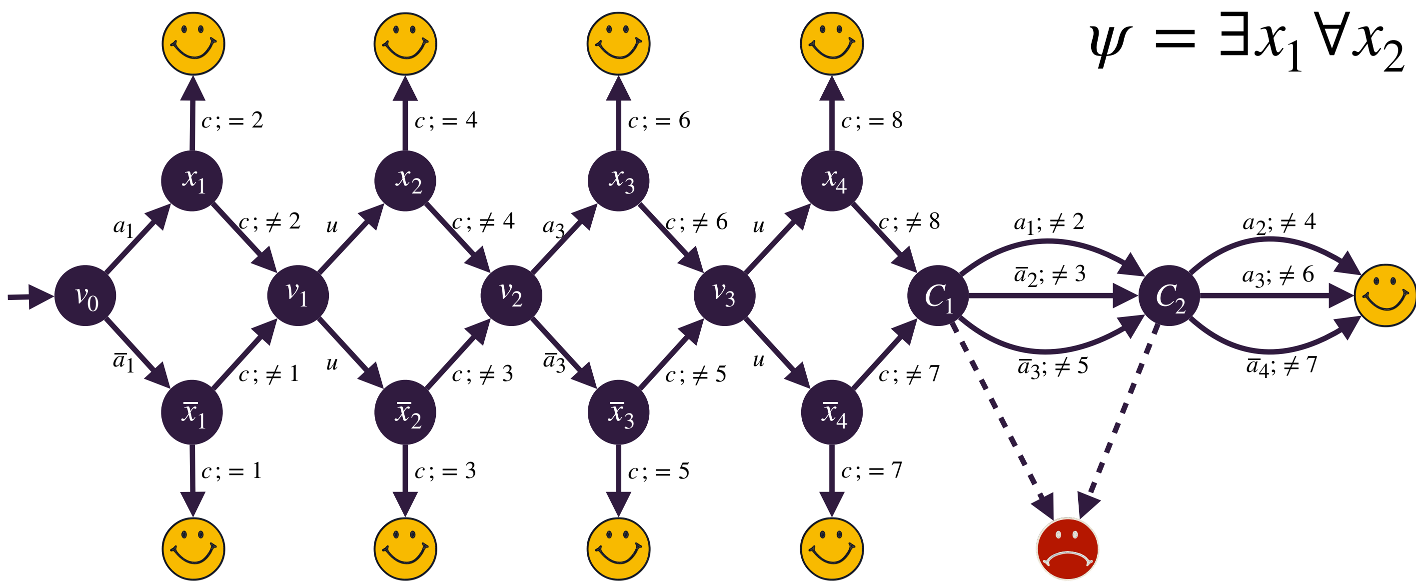


$$\psi = \exists x_1 \forall x_2 \exists x_3 \forall x_4 \cdot (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_2 \vee x_3 \vee \neg x_4)$$

Strategy for Gru if ψ is true

- At v_0, play the correct assignment, say false (i.e. $\overline{a}_1$), reaching $\overline{x}_1$
  - If $k = 1$, then go to 
  - If $k \neq 1$, the game proceeds to v_1

PSPACE-hardness - 2

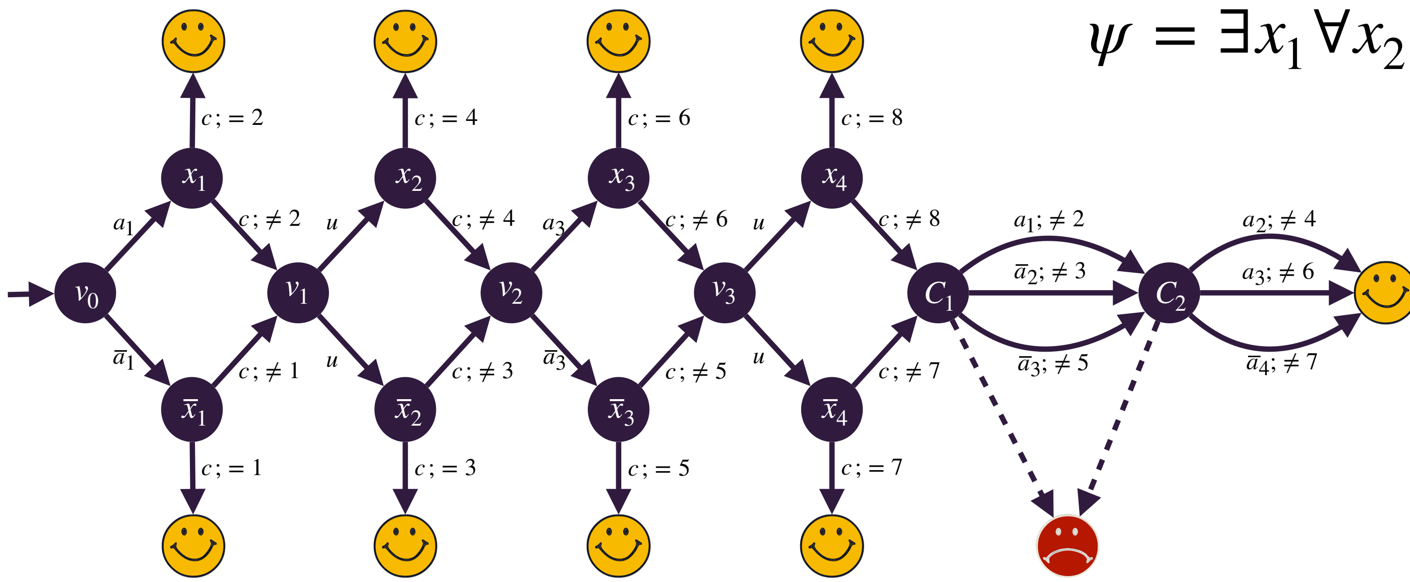


$$\psi = \exists x_1 \forall x_2 \exists x_3 \forall x_4 \cdot (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_2 \vee x_3 \vee \neg x_4)$$

Strategy for Gru if ψ is true

- At v_0, play the correct assignment, say false (i.e. $\overline{a}_1$), reaching $\overline{x}_1$
  - If $k = 1$, then go to 
  - If $k \neq 1$, the game proceeds to v_1
- At v_1, play u (no choice — the next vertex is then choose non-deterministically):

PSPACE-hardness - 2

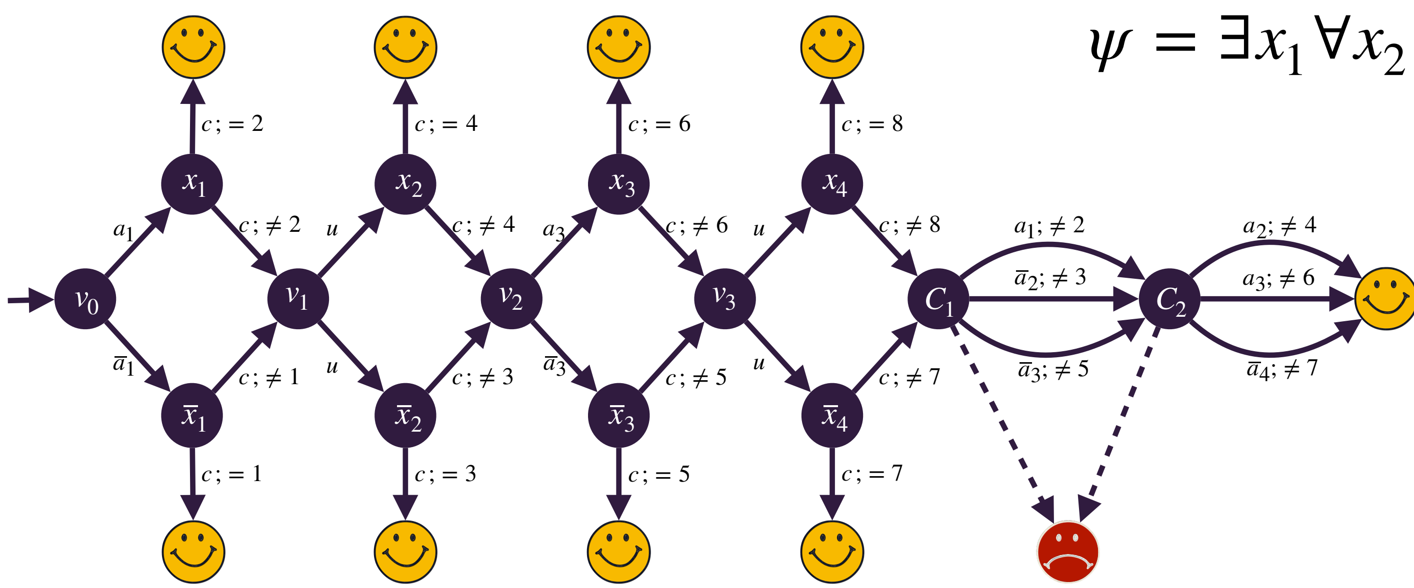


$$\psi = \exists x_1 \forall x_2 \exists x_3 \forall x_4 \cdot (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_2 \vee x_3 \vee \neg x_4)$$

Strategy for Gru if ψ is true

- At v_0, play the correct assignment, say false (i.e. $\overline{a}_1$), reaching $\overline{x}_1$
  - If $k = 1$, then go to
  - If $k \neq 1$, the game proceeds to v_1
- At v_1, play u (no choice — the next vertex is then choose non-deterministically):
  - Either the game proceeds to x_2 (encoding true), and if $k = 4$, then go to

PSPACE-hardness - 2

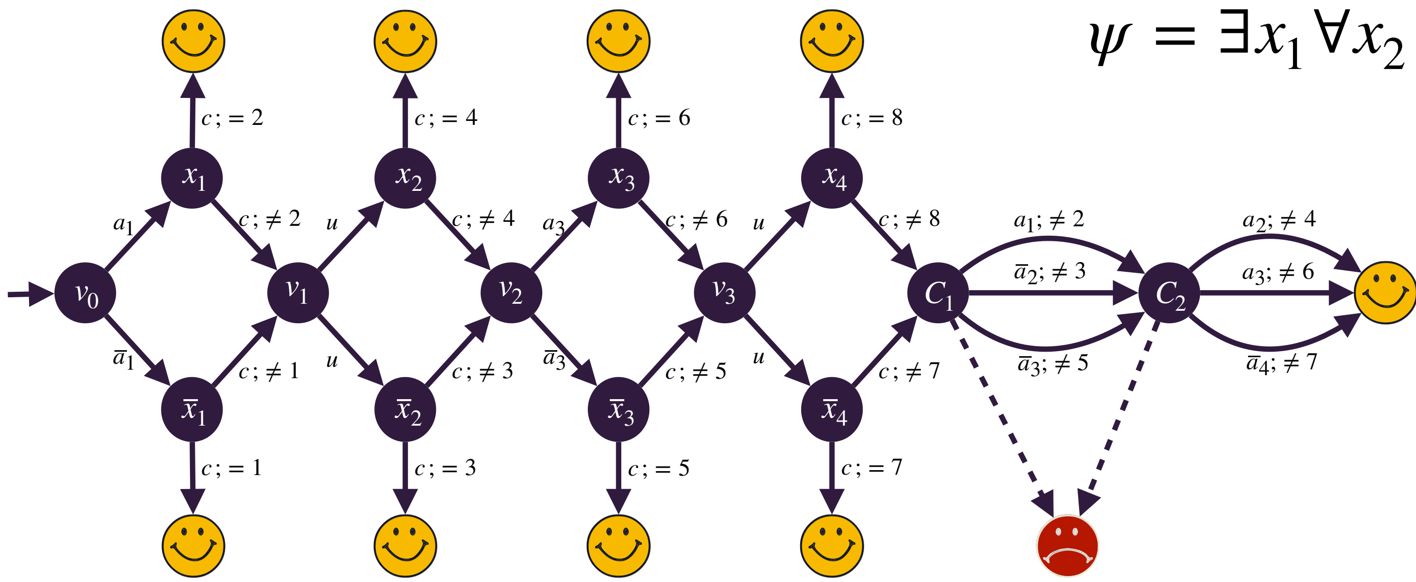


$\psi = \exists x_1 \forall x_2 \exists x_3 \forall x_4 \cdot (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_2 \vee x_3 \vee \neg x_4)$

Strategy for Gru if ψ is true

- At v_0, play the correct assignment, say false (i.e. $\bar{a}_1$), reaching $\bar{x}_1$
  - If $k = 1$, then go to ☺
  - If $k \neq 1$, the game proceeds to v_1
- At v_1, play u (no choice — the next vertex is then choose non-deterministically):
  - Either the game proceeds to x_2 (encoding true), and if $k = 4$, then go to ☺
  - Or the game proceeds to $\bar{x}_2$ (encoding false), and if $k = 3$, then go to ☺

PSPACE-hardness - 2



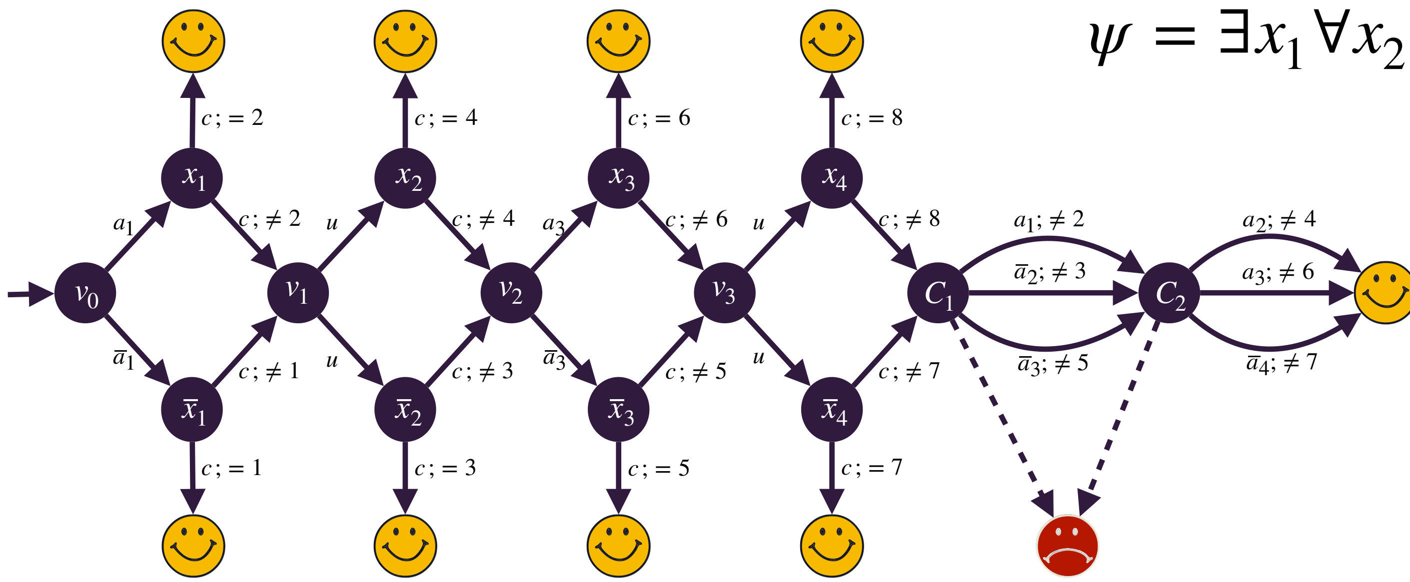
$$\psi = \exists x_1 \forall x_2 \exists x_3 \forall x_4 \cdot (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_2 \vee x_3 \vee \neg x_4)$$

Strategy for Gru if ψ is true

- At v_0, play the correct assignment, say false (i.e. $\overline{a}_1$), reaching $\overline{x}_1$
  - If $k = 1$, then go to 
  - If $k \neq 1$, the game proceeds to v_1
- At v_1, play u (no choice — the next vertex is then choose non-deterministically):
  - Either the game proceeds to x_2 (encoding true), and if $k = 4$, then go to 
  - Or the game proceeds to $\overline{x}_2$ (encoding false), and if $k = 3$, then go to 
  - Otherwise the game proceeds to v_2

PSPACE-hardness - 2

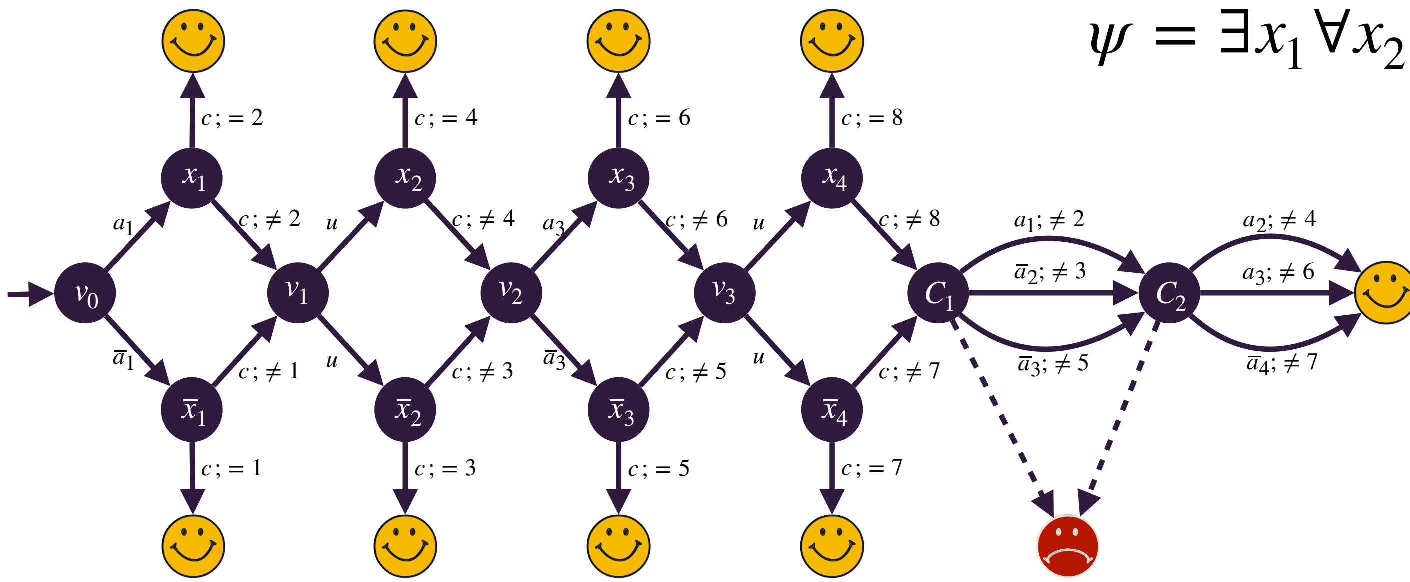
$$\psi = \exists x_1 \forall x_2 \exists x_3 \forall x_4 \cdot (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_2 \vee x_3 \vee \neg x_4)$$



Strategy for Gru if ψ is true

- At v_0, play the correct assignment, say false (i.e. $\overline{a}_1$), reaching $\overline{x}_1$
  - If $k = 1$, then go to
  - If $k \neq 1$, the game proceeds to v_1
- At v_1, play u (no choice — the next vertex is then choose non-deterministically):
  - Either the game proceeds to x_2 (encoding true), and if $k = 4$, then go to
  - Or the game proceeds to $\overline{x}_2$ (encoding false), and if $k = 3$, then go to
  - Otherwise the game proceeds to v_2
- Etc...

PSPACE-hardness - 2



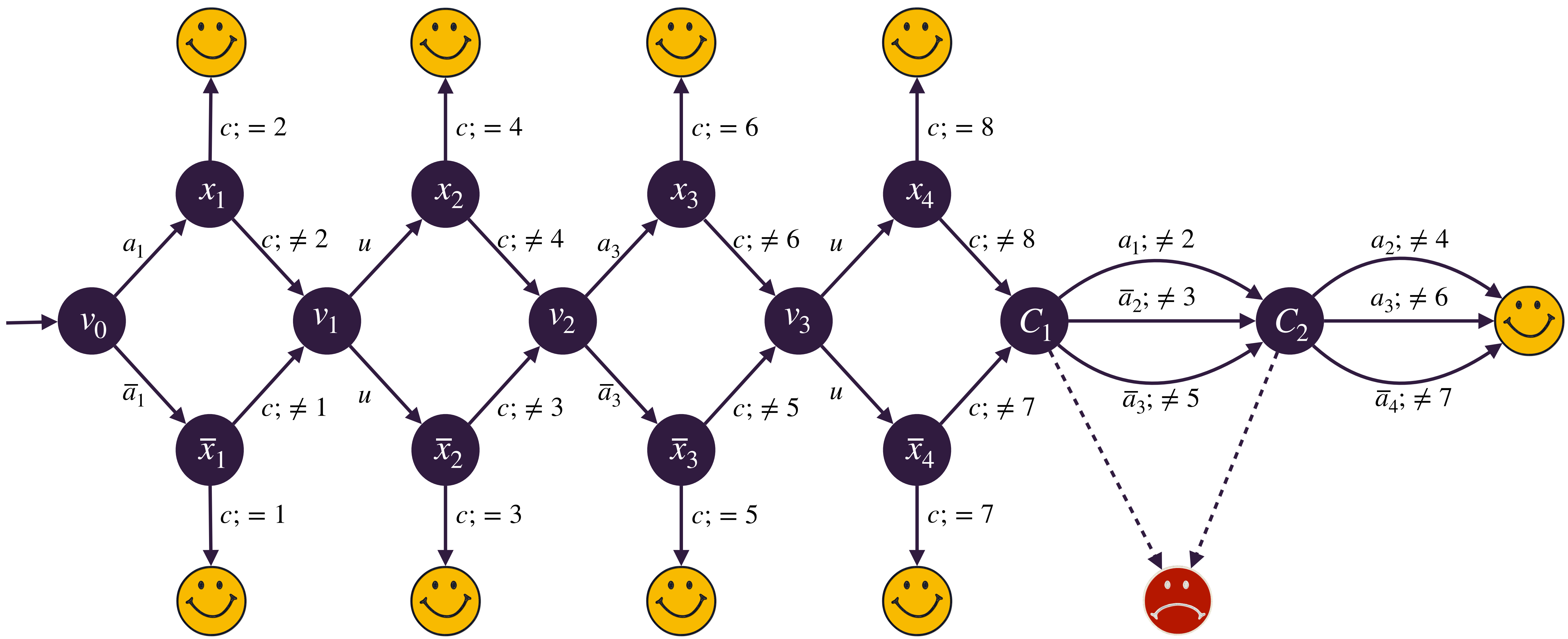
$$\psi = \exists x_1 \forall x_2 \exists x_3 \forall x_4 \cdot (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_2 \vee x_3 \vee \neg x_4)$$

Strategy for Gru if ψ is true

- At v_0, play the correct assignment, say false (i.e. $\overline{a}_1$), reaching $\overline{x}_1$
  - If $k = 1$, then go to 
  - If $k \neq 1$, the game proceeds to v_1
- At v_1, play u (no choice — the next vertex is then choose non-deterministically):
  - Either the game proceeds to x_2 (encoding true), and if $k = 4$, then go to 
  - Or the game proceeds to $\overline{x}_2$ (encoding false), and if $k = 3$, then go to 
  - Otherwise the game proceeds to v_2
- Etc…
- At C_1, $k \neq 1$, $k \neq 4$ (in the first case above), …. hence Gru can enforce  if and only if the chosen assignment makes the two clauses true

PSPACE-hardness - 3

QSAT formula $\psi = \exists x_1 \forall x_2 \exists x_3 \forall x_4 \cdot (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_2 \vee x_3 \vee \neg x_4)$



ψ is true iff Gru has a winning strategy in the above counting game

Going further?

Going further?

- The previous approach yielding the PSPACE upper bound applies to many other Boolean objectives, as long as solving the corresponding standard games can be achieved in PSPACE

Going further?

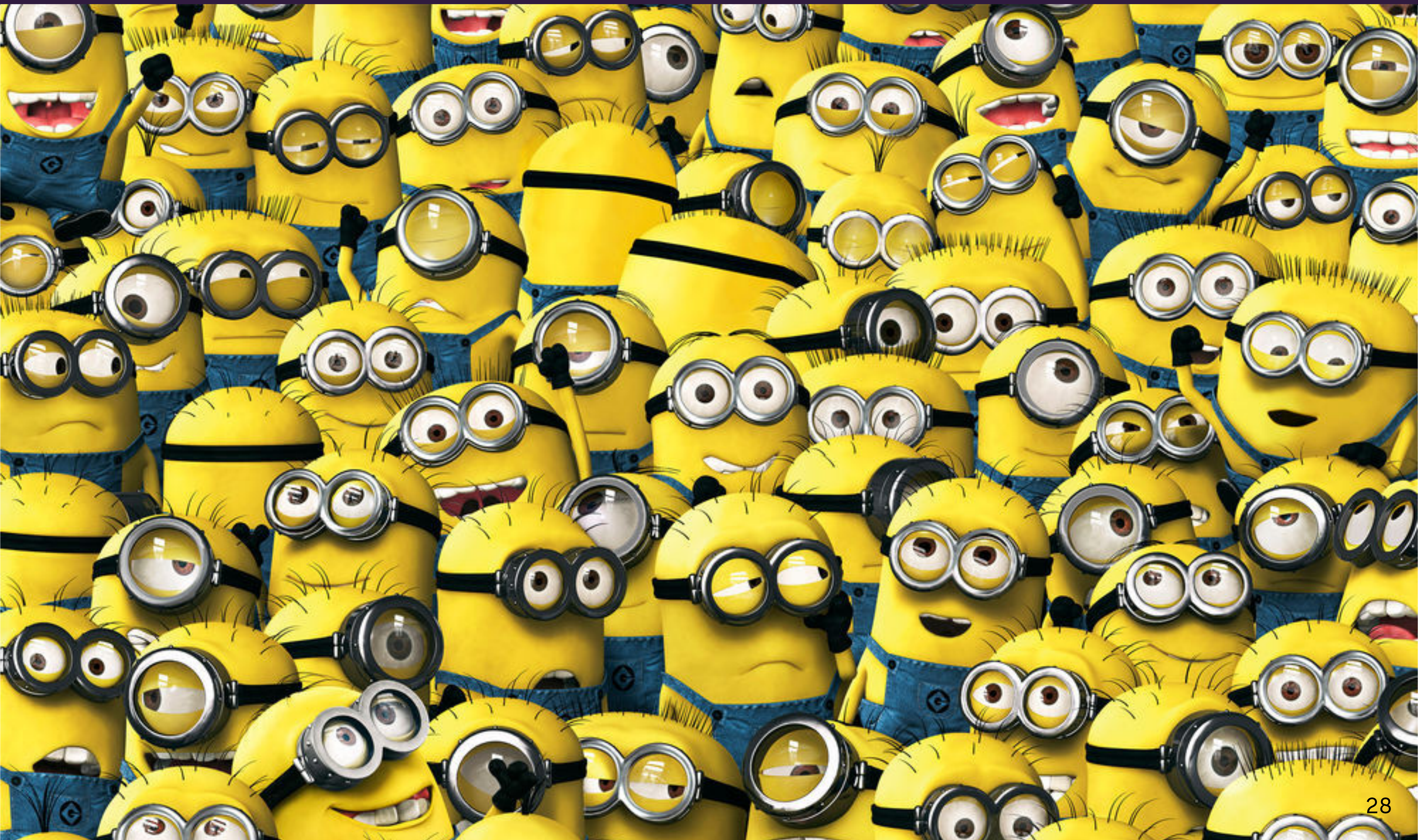
- The previous approach yielding the PSPACE upper bound applies to many other Boolean objectives, as long as solving the corresponding standard games can be achieved in PSPACE

- What about more involved quantitative objectives/payoffs?

Going further?

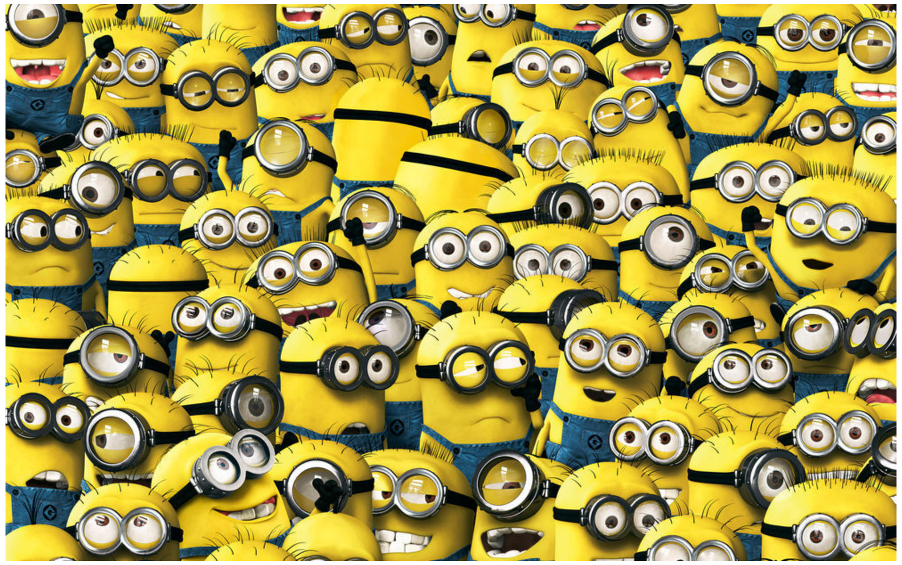
- The previous approach yielding the PSPACE upper bound applies to many other Boolean objectives, as long as solving the corresponding standard games can be achieved in PSPACE
- What about more involved quantitative objectives/payoffs?
- We believe the approach can be extended to « structured » infinite-state systems (e.g. pushdown systems)

The coalition problem

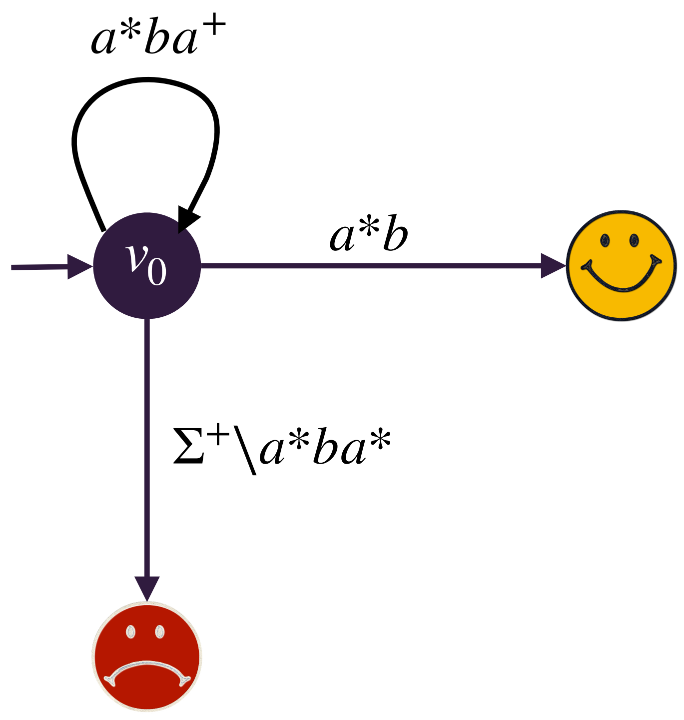


The coalition problem

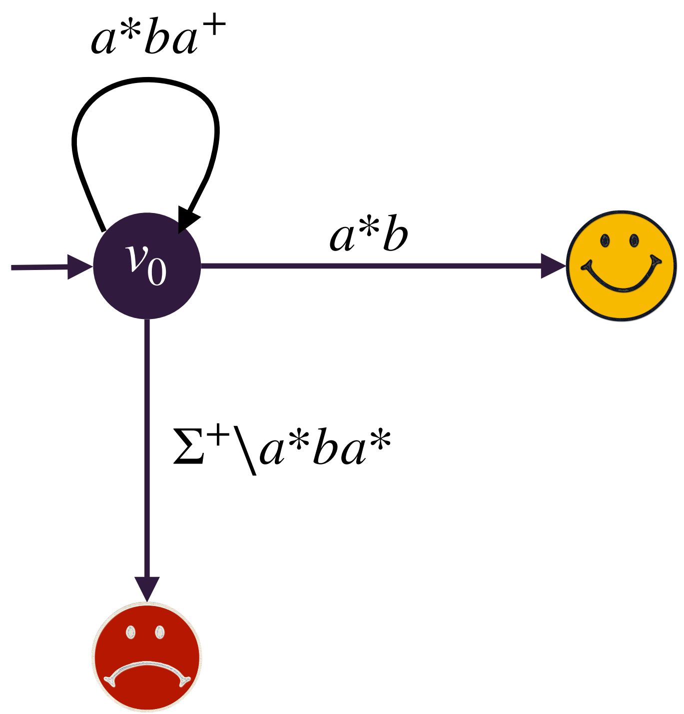
- Input: parameterized game $G = (V, \delta)$ and linear property φ
- Question: does there exist $(\sigma_i)_{i \geq 1}$ such that for every k, for every $\rho \in \mathsf{Out}\big((\sigma_i)_{1 \leq i \leq k}\big)$, $\rho \models \varphi$?



An intriguing example



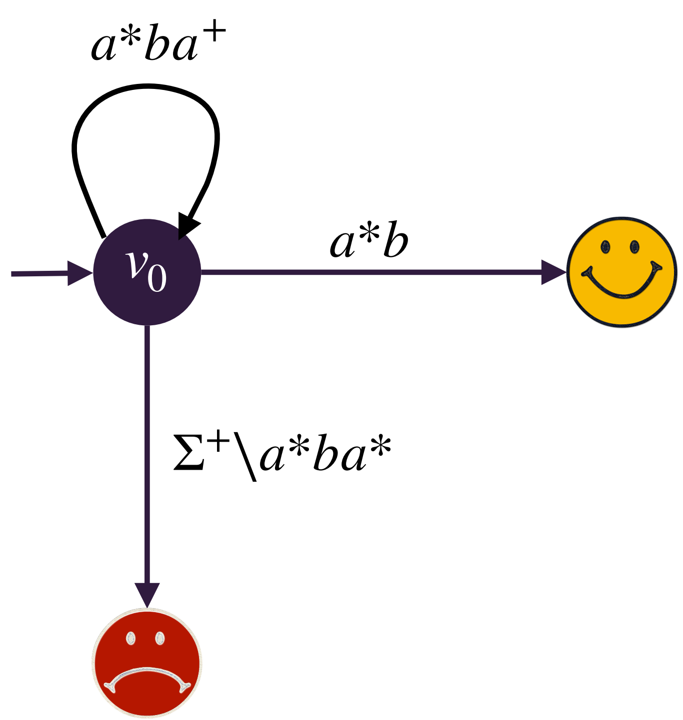
An intriguing example



A winning coalition strategy

- At round i:
  - Player i plays b
  - Player $j \neq i$ plays a

An intriguing example



A winning coalition strategy

- At round i:
  - Player i plays b
  - Player $j \neq i$ plays a

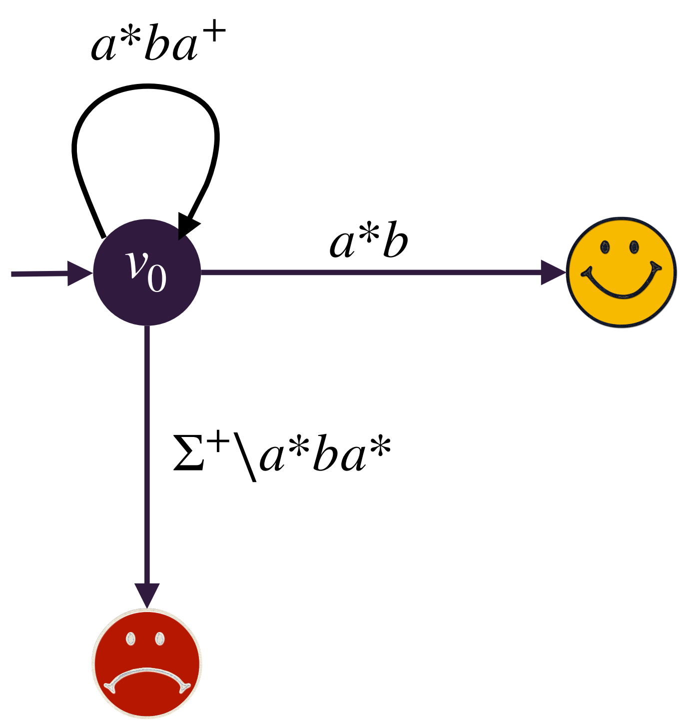
If $k = 1$: $v_0 \xrightarrow{b}$ ☺

If $k = 2$: $v_0 \xrightarrow{ba} v_0 \xrightarrow{ab}$ ☺

If $k = 3$: $v_0 \xrightarrow{baa} v_0 \xrightarrow{aba} v_0 \xrightarrow{aab}$ ☺

⋮

An intriguing example



A winning coalition strategy

- At round i:
  - Player i plays b
  - Player $j \neq i$ plays a
- At round i, coalition plays $a^{i-1}ba^{\omega}$

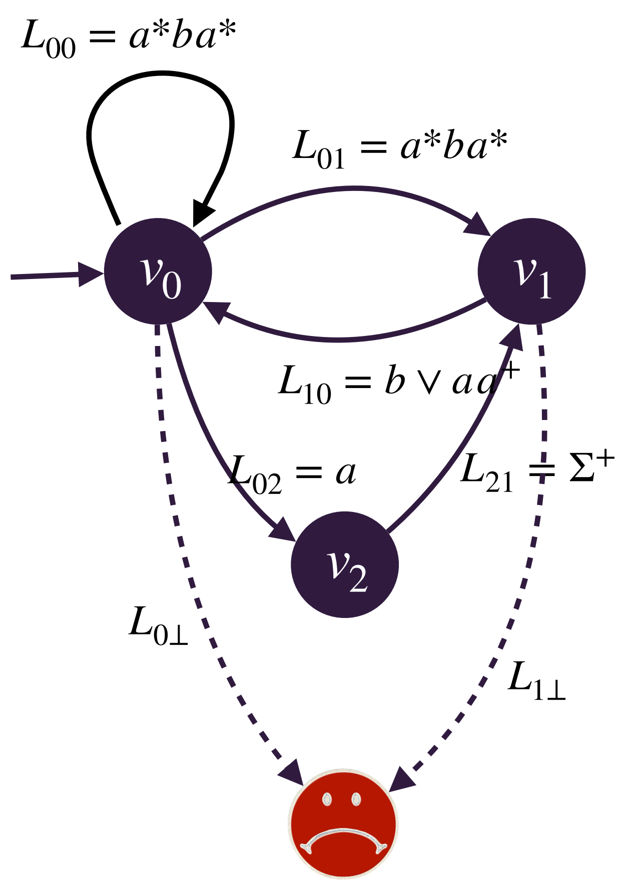
If $k = 1$: $v_0 \xrightarrow{b}$ ☺

If $k = 2$: $v_0 \xrightarrow{ba} v_0 \xrightarrow{ab}$ ☺

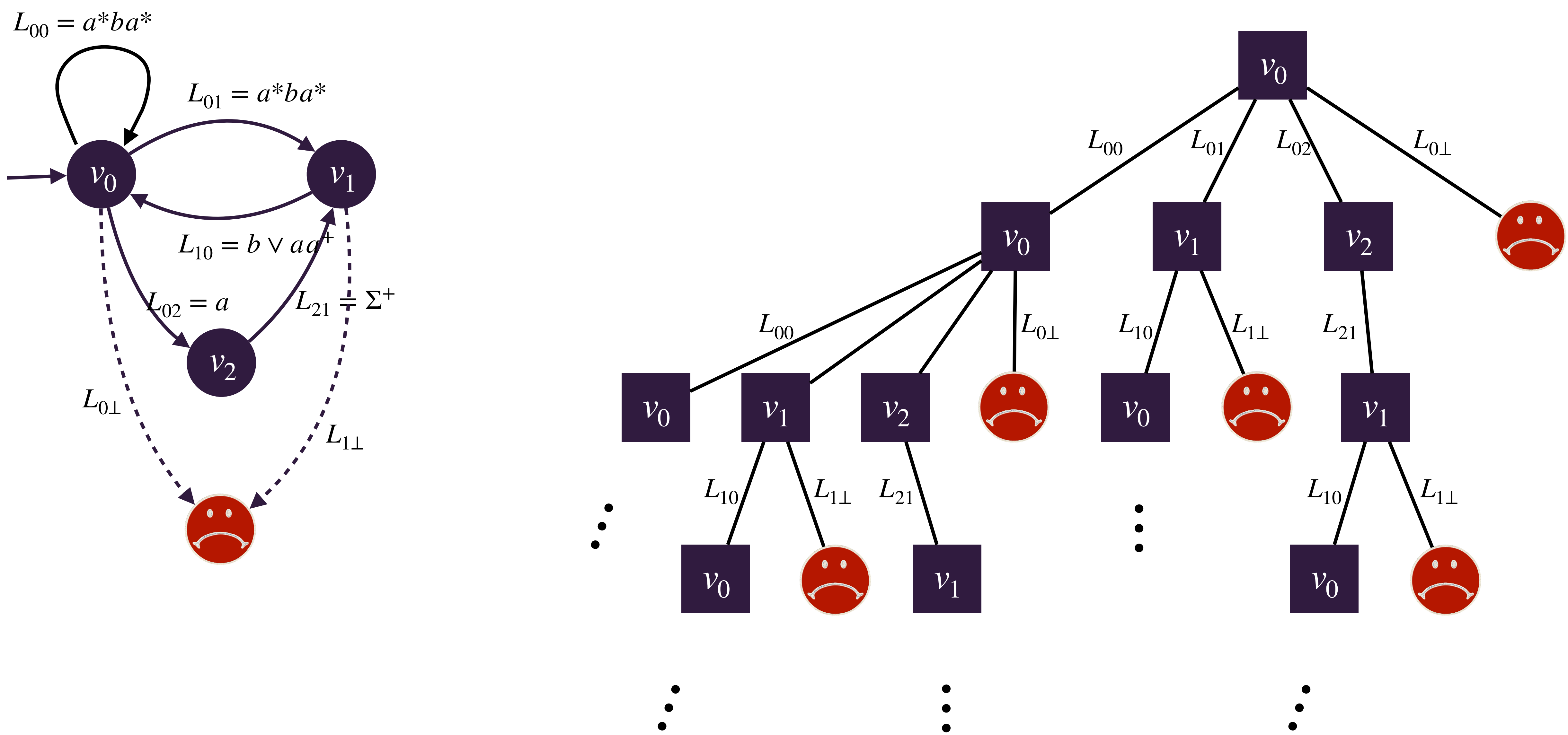
If $k = 3$: $v_0 \xrightarrow{baa} v_0 \xrightarrow{aba} v_0 \xrightarrow{aab}$ ☺

$\vdots$

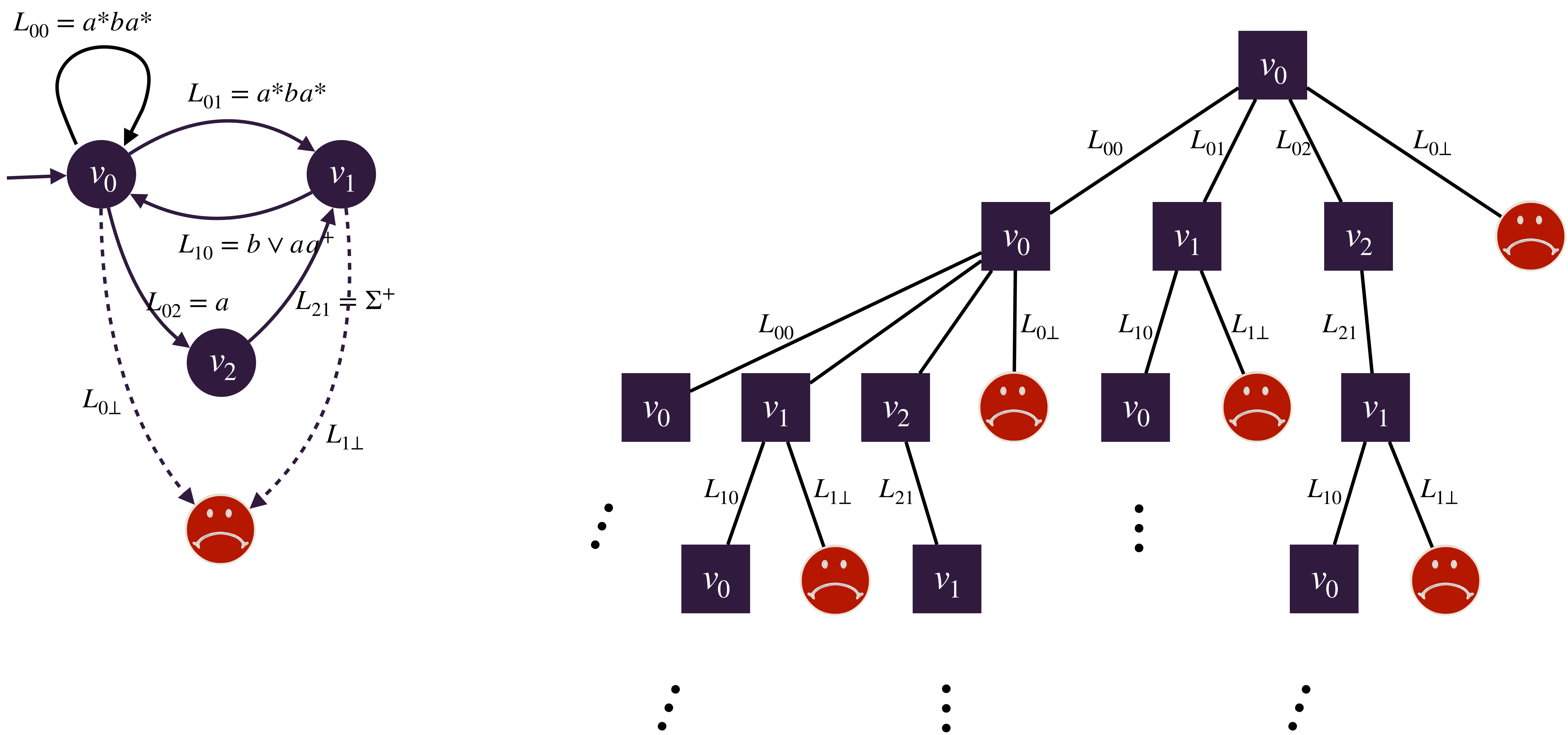
Tree unfolding



Tree unfolding

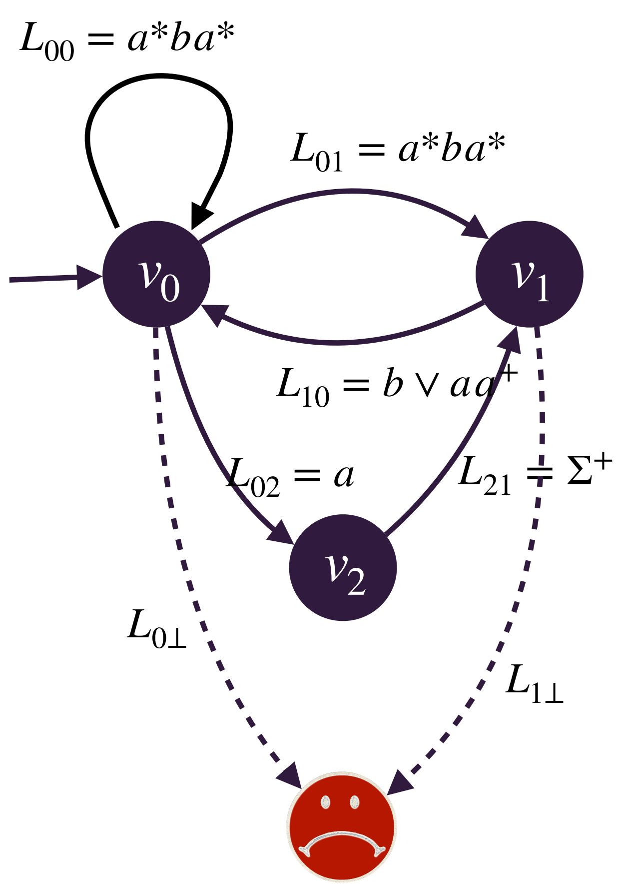


Tree unfolding



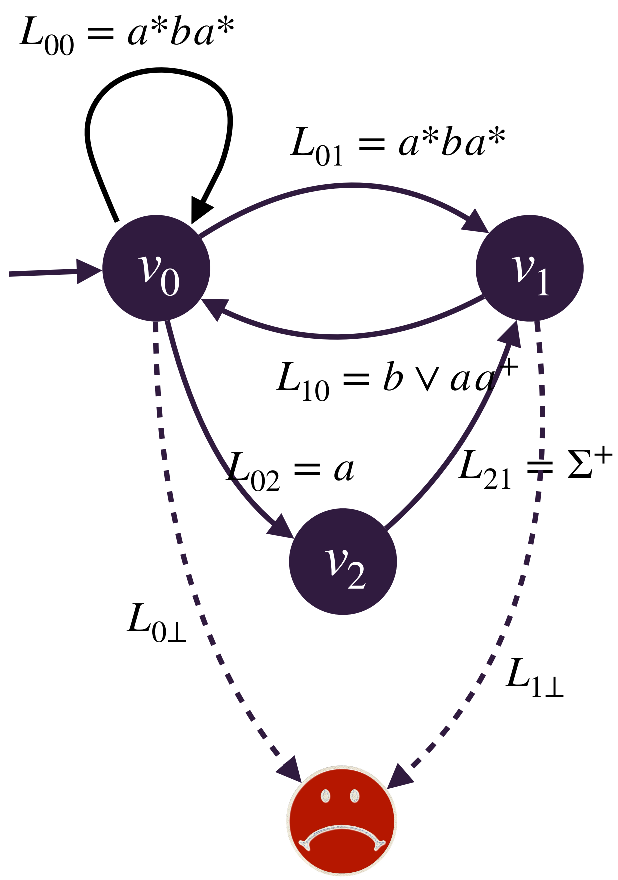
There is a winning coalition strategy in the game iff there is a winning coalition strategy in the unfolding

Tree unfolding — Safety case



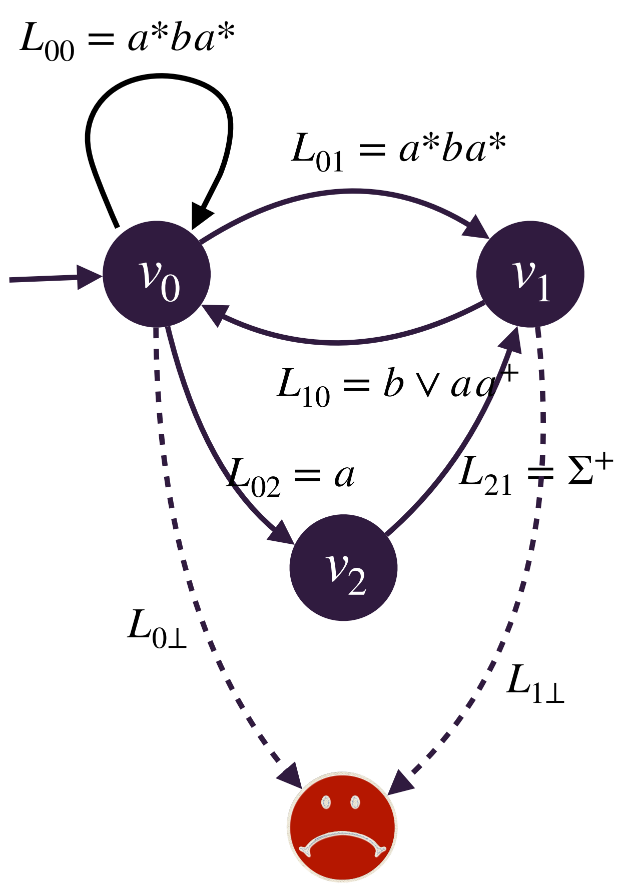
Tree unfolding — Safety case

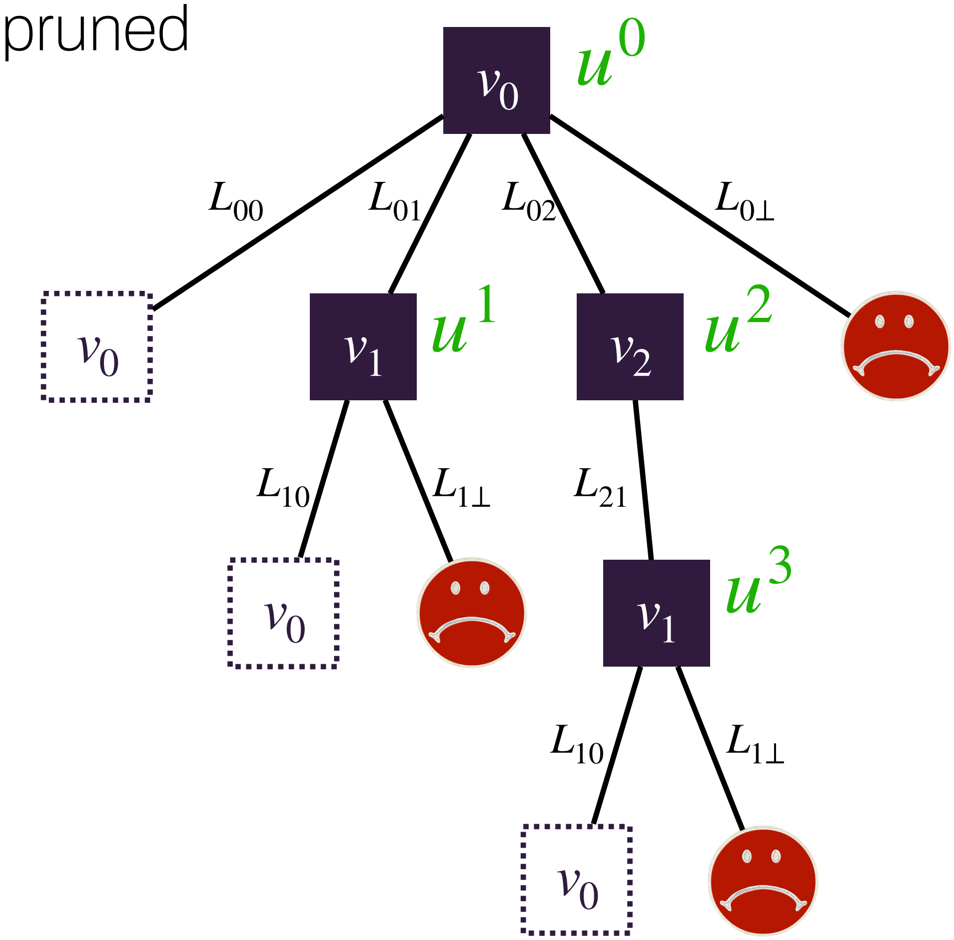
For a safety condition: the unfolding can be pruned



Tree unfolding — Safety case

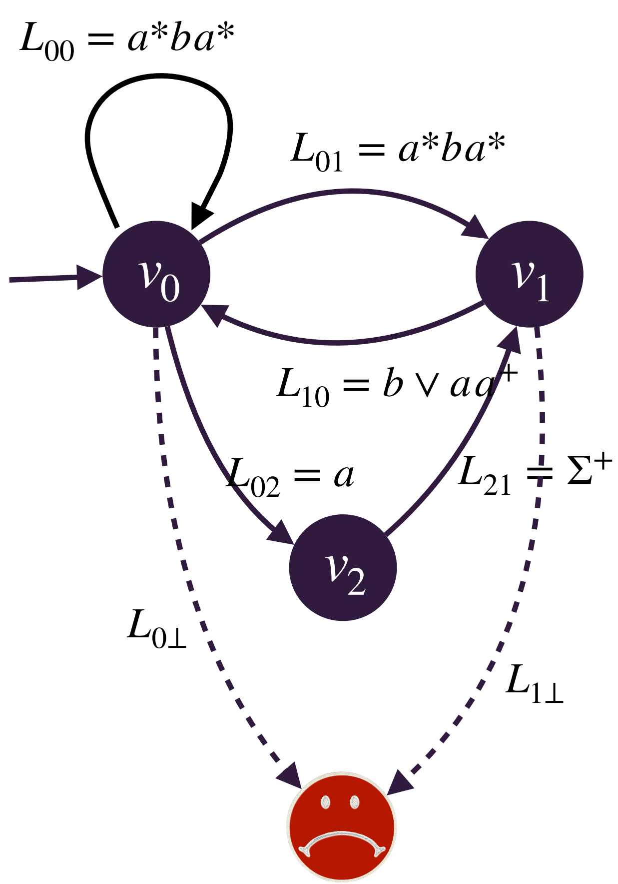
For a safety condition: the unfolding can be pruned





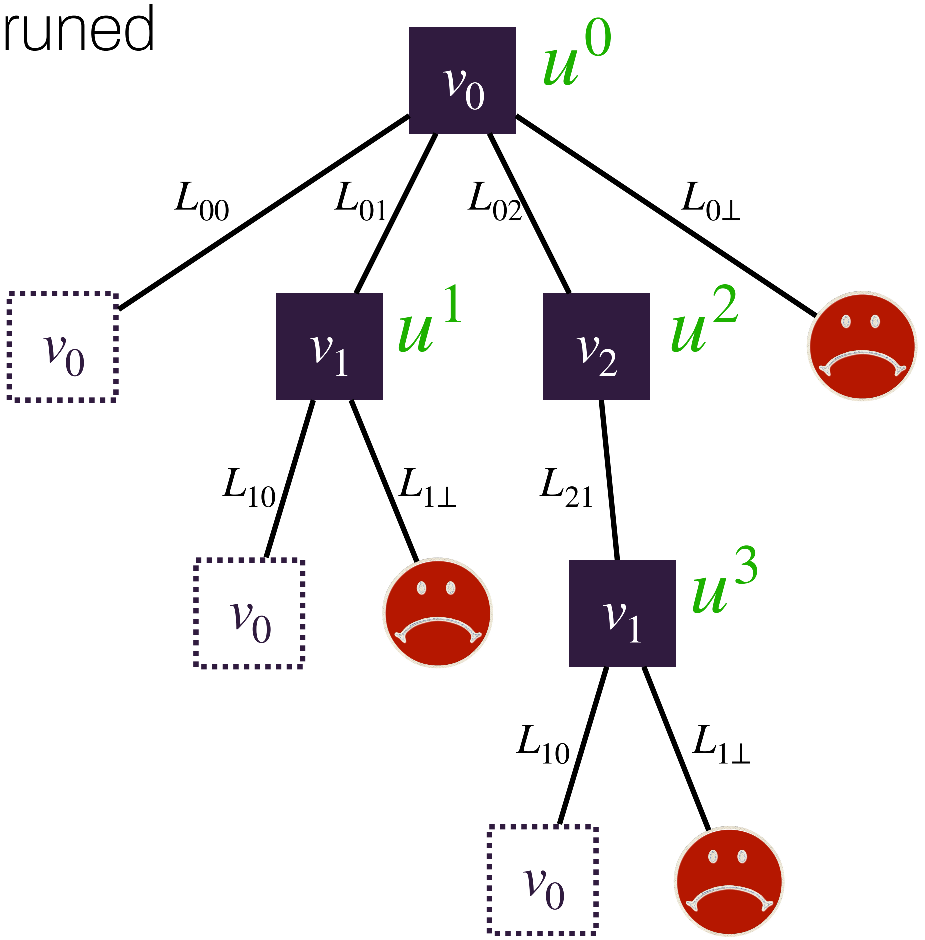
Tree unfolding — Safety case

For a safety condition: the unfolding can be pruned



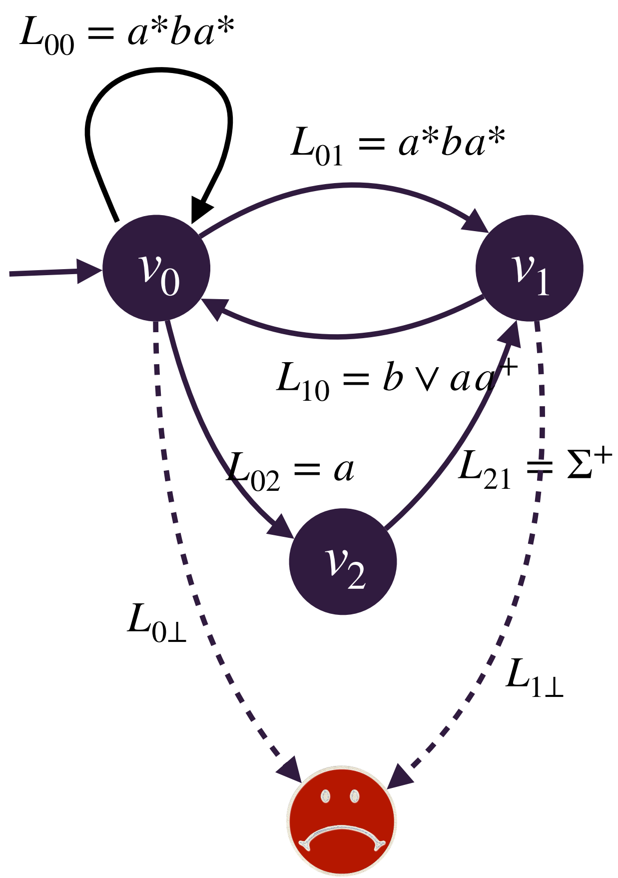
Possible solution:

- $u^0 = aba^\omega$
- $u^1 = a^\omega$
- $u^2 = a^\omega$
- $u^3 = ba^\omega$



Tree unfolding — Safety case

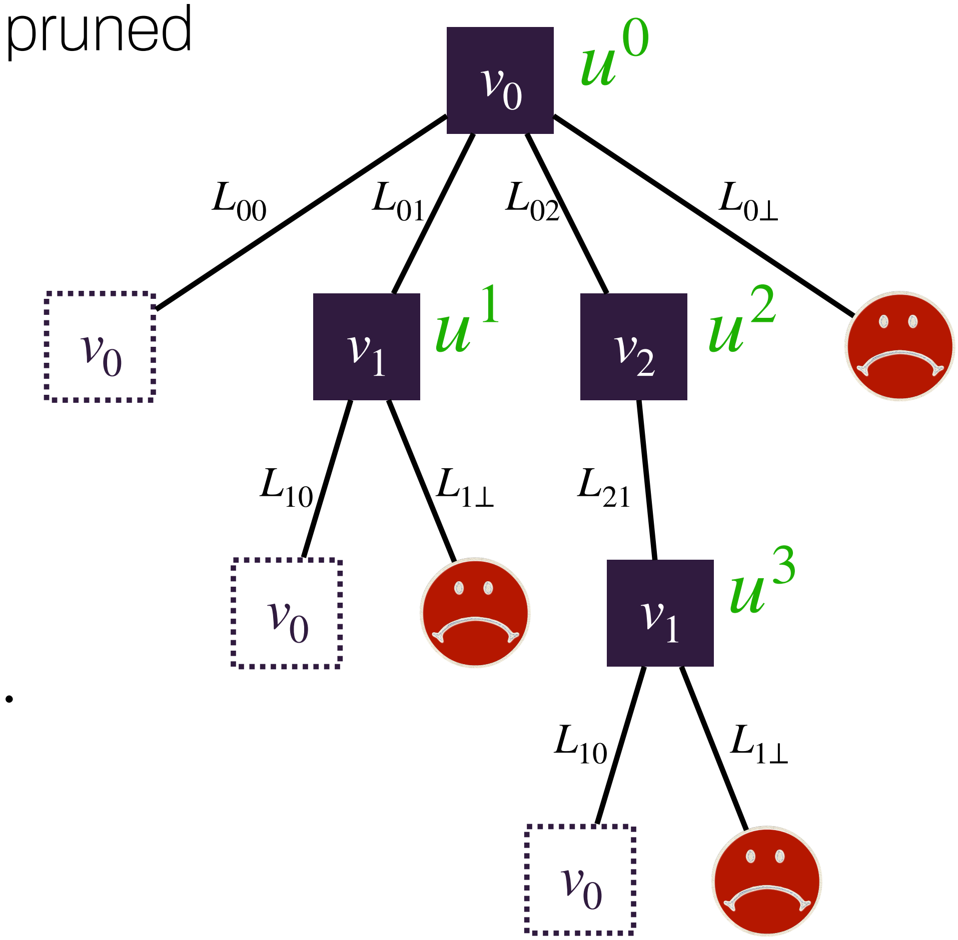
For a safety condition: the unfolding can be pruned



Possible solution:

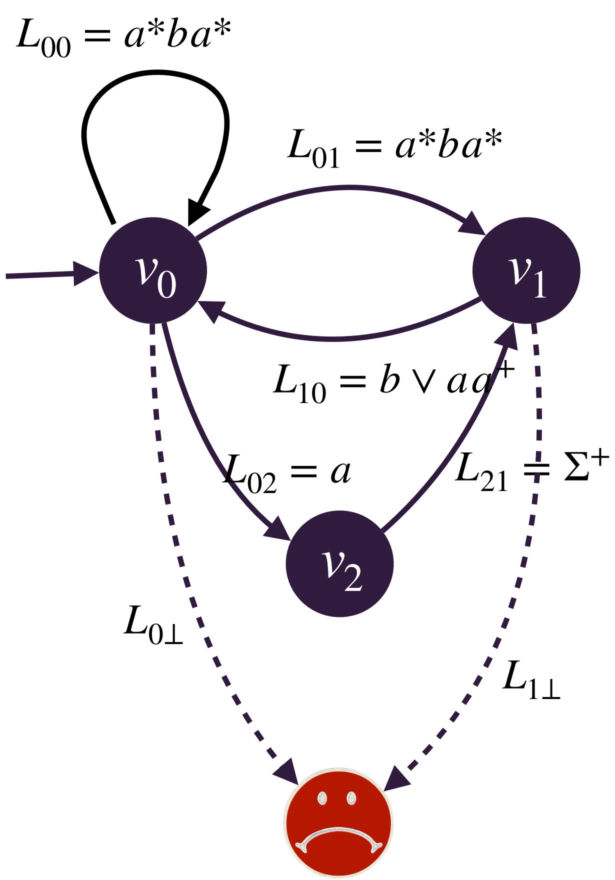
- $u^0 = aba^\omega$
- $u^1 = a^\omega$
- $u^2 = a^\omega$
- $u^3 = ba^\omega$

▸ $k = 1$: $v_0 \xrightarrow{a} v_2 \xrightarrow{a} v_1 \xrightarrow{b} v_0 \to \ldots$
since $a \in L_{02}$, $a \in L_{21}$, $b \in L_{10}$



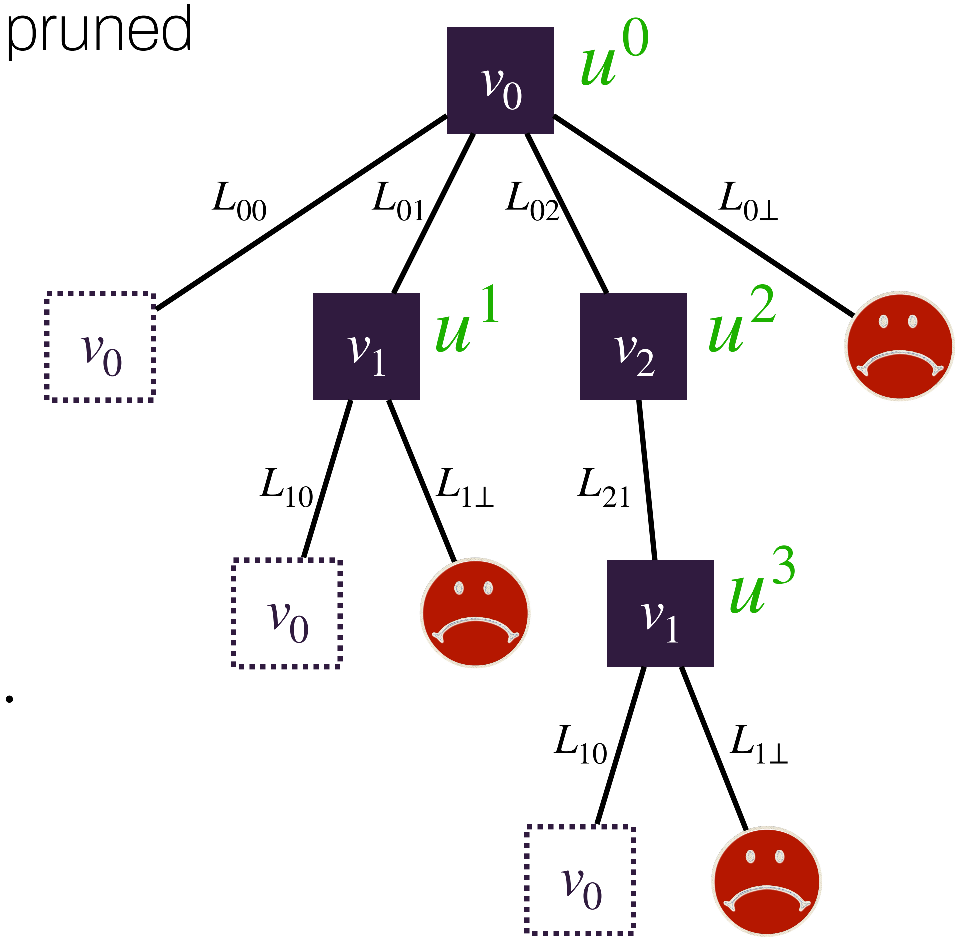
Tree unfolding — Safety case

For a safety condition: the unfolding can be pruned



Possible solution:

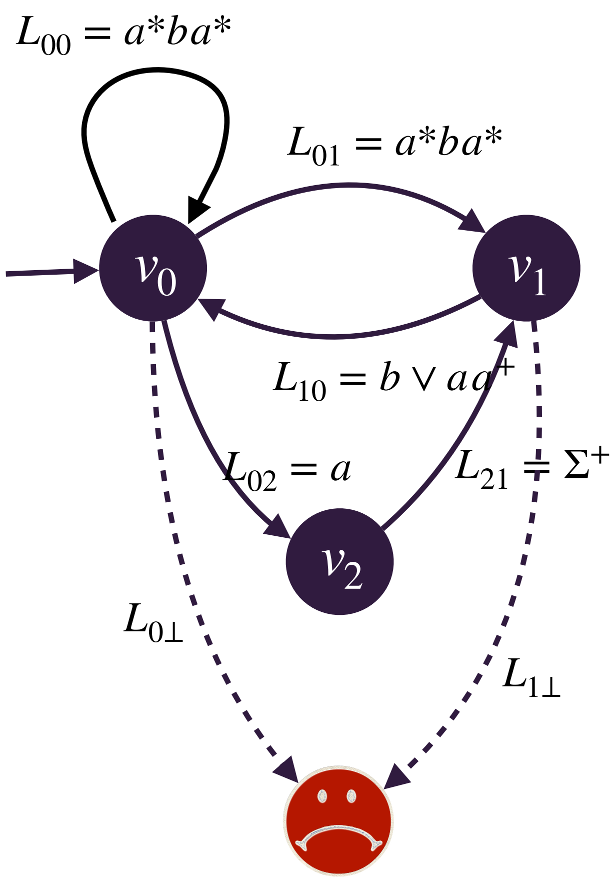
- $u^0 = aba^\omega$
- $u^1 = a^\omega$
- $u^2 = a^\omega$
- $u^3 = ba^\omega$



- $k = 1$: $v_0 \xrightarrow{a} v_2 \xrightarrow{a} v_1 \xrightarrow{b} v_0 \rightarrow \ldots$
  since $a \in L_{02}$, $a \in L_{21}$, $b \in L_{10}$
- $k > 1$: $v_0 \xrightarrow{aba^{k-2}} v_1 \xrightarrow{a^k} v_0 \rightarrow \ldots$
  since $aba^{k-2} \in L_{01}$, $a^k \in L_{10}$

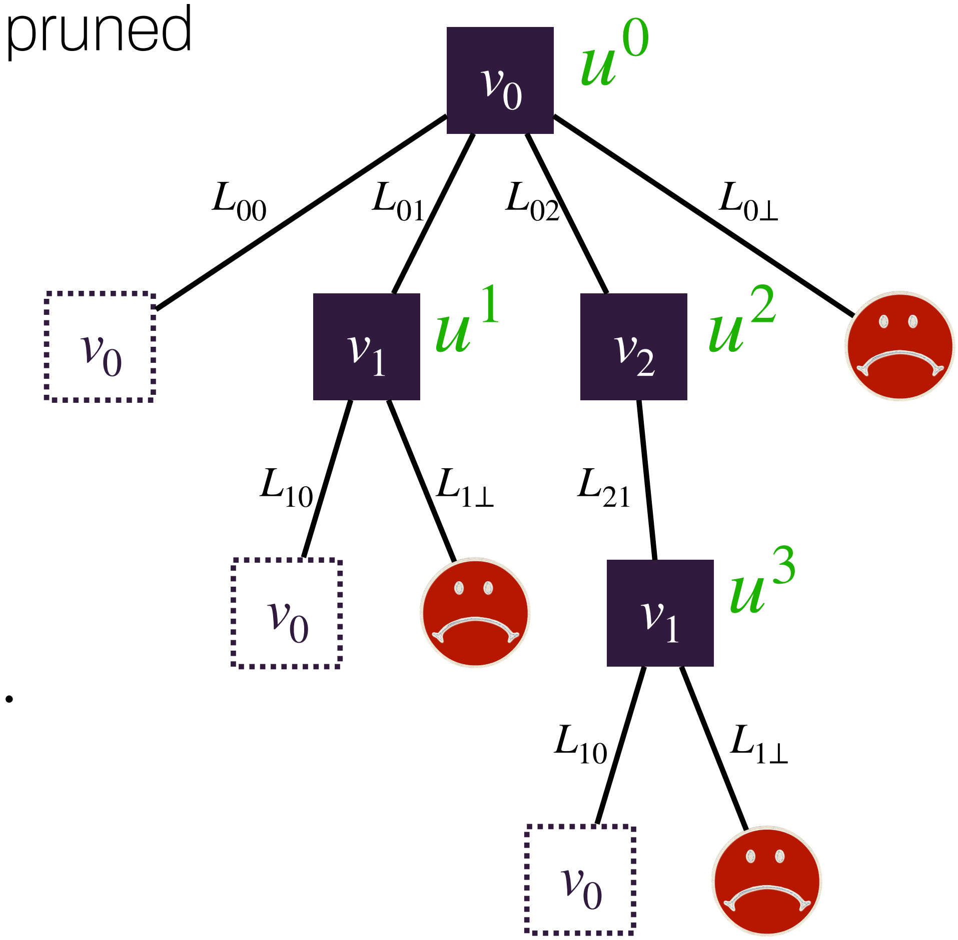
Tree unfolding — Safety case

For a safety condition: the unfolding can be pruned



Possible solution:

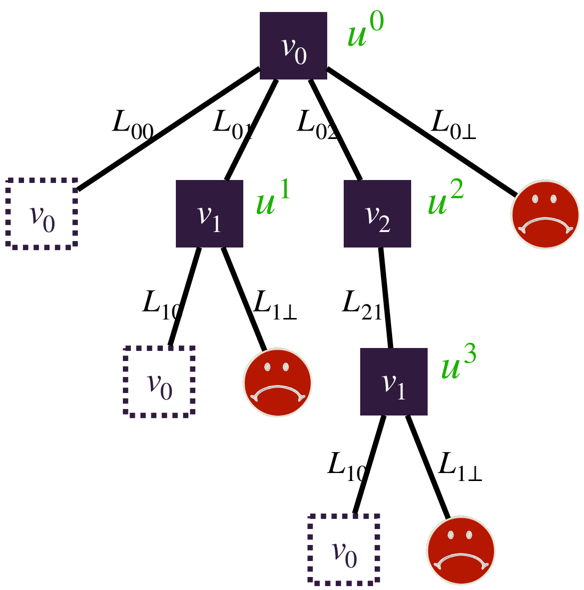
- $u^0 = aba^\omega$
- $u^1 = a^\omega$
- $u^2 = a^\omega$
- $u^3 = ba^\omega$



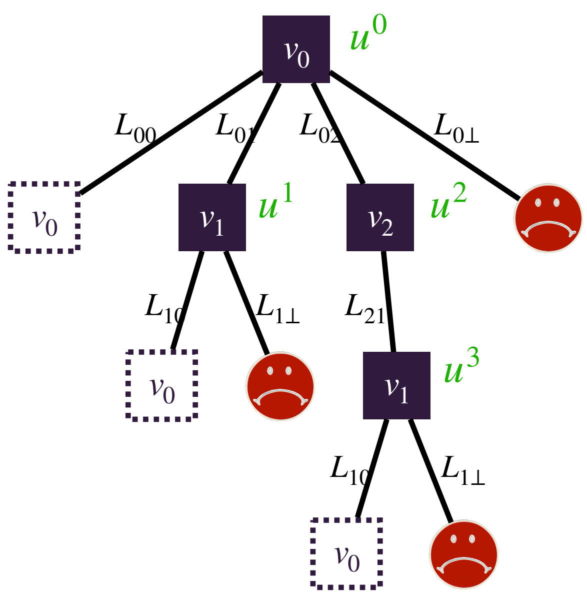
- $k = 1$: $v_0 \xrightarrow{a} v_2 \xrightarrow{a} v_1 \xrightarrow{b} v_0 \rightarrow \ldots$
  since $a \in L_{02}$, $a \in L_{21}$, $b \in L_{10}$
- $k > 1$: $v_0 \xrightarrow{aba^{k-2}} v_1 \xrightarrow{a^k} v_0 \rightarrow \ldots$
  since $aba^{k-2} \in L_{01}$, $a^k \in L_{10}$

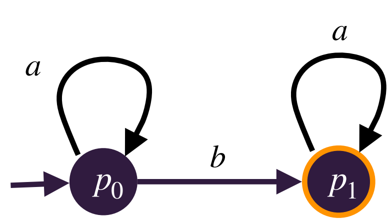
There is a winning coalition strategy in the unfolding iff there are infinite words $(u^i)_i$ s.t. for every $k \geq 1$, playing $u^i_{\leq k}$ at each internal node ensures avoiding ☹

Construction of a finite automaton — 1



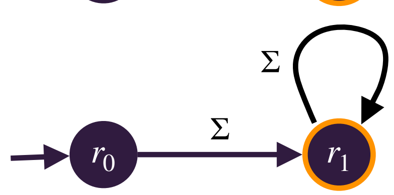
Construction of a finite automaton — 1



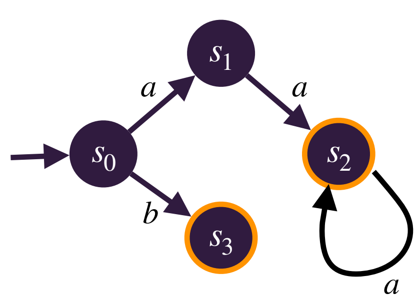


$L_{00} = L_{01}$

L_{02}

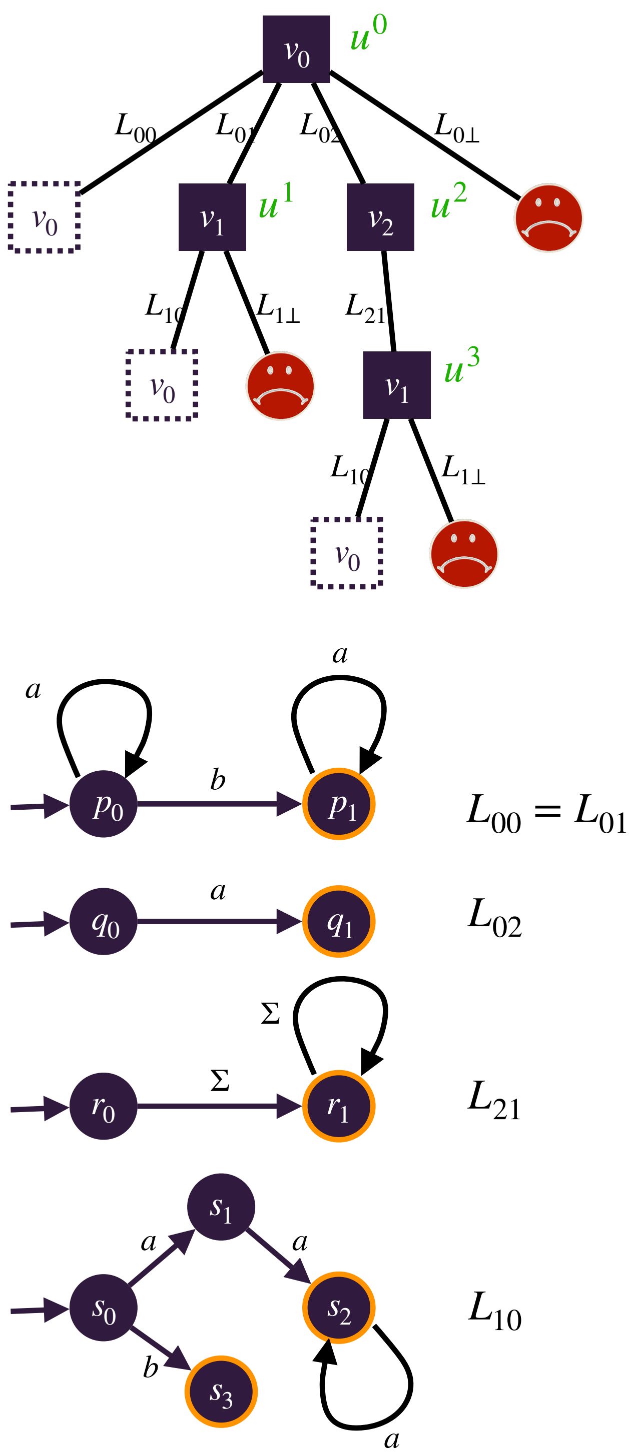


L_{21}

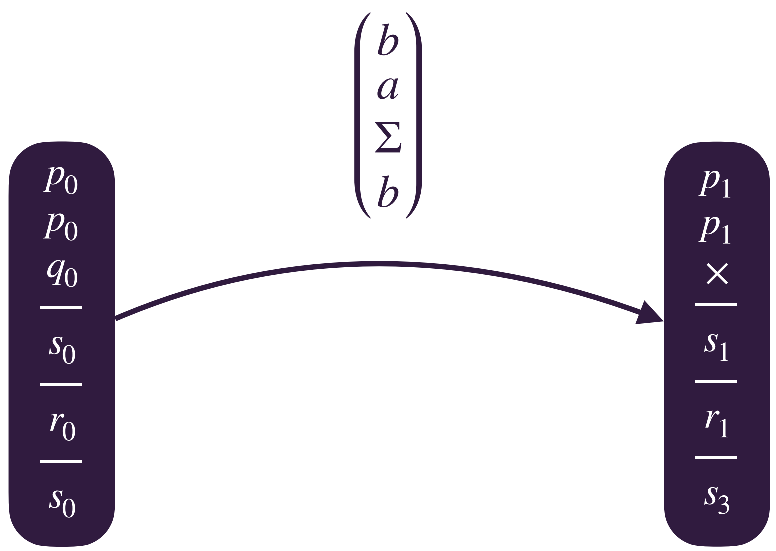


L_{10}

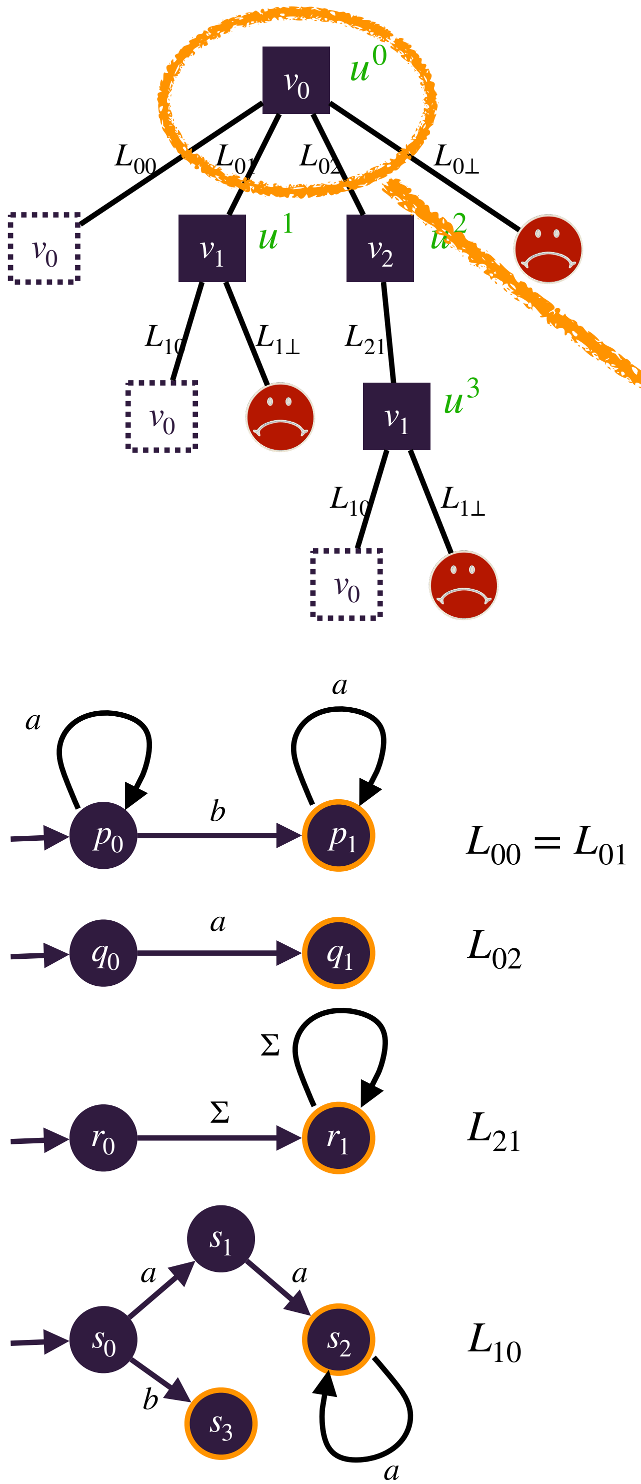
Construction of a finite automaton — 1



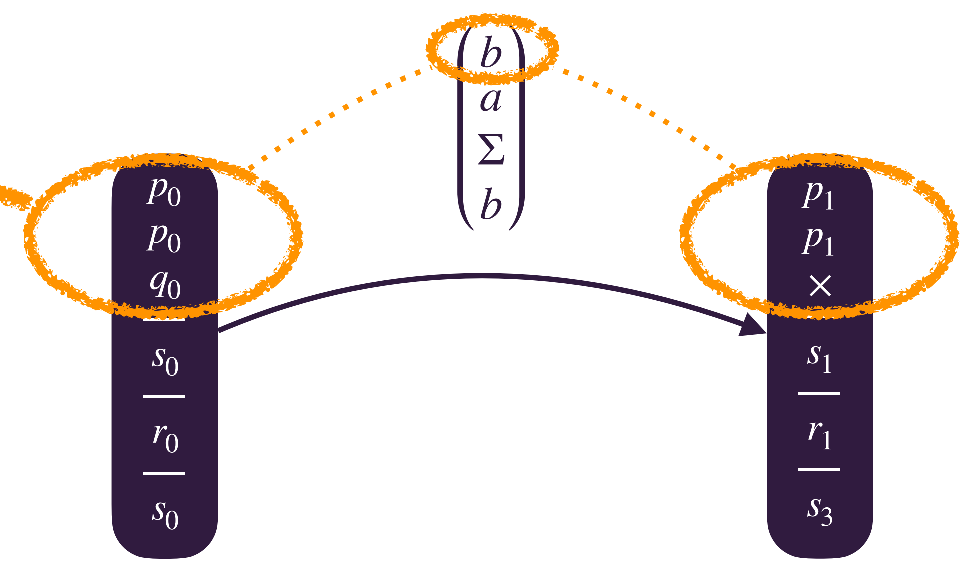
We launch several computations in parallel:



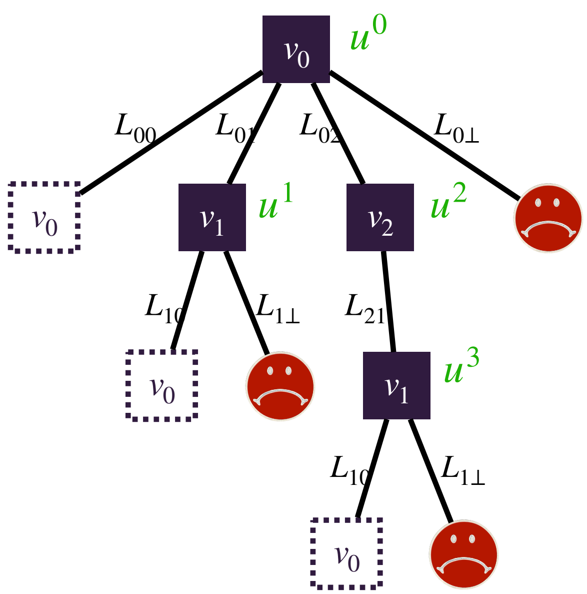
Construction of a finite automaton — 1

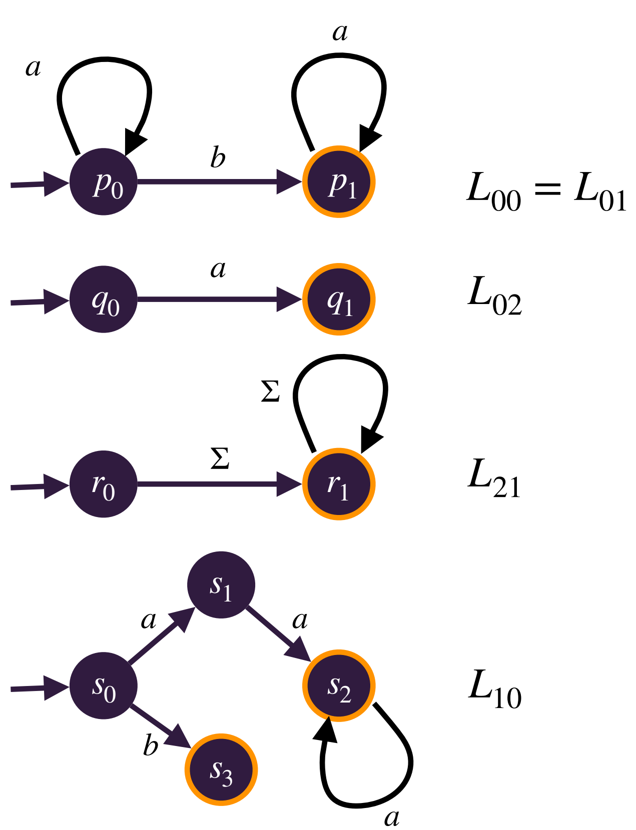


We launch several computations in parallel:

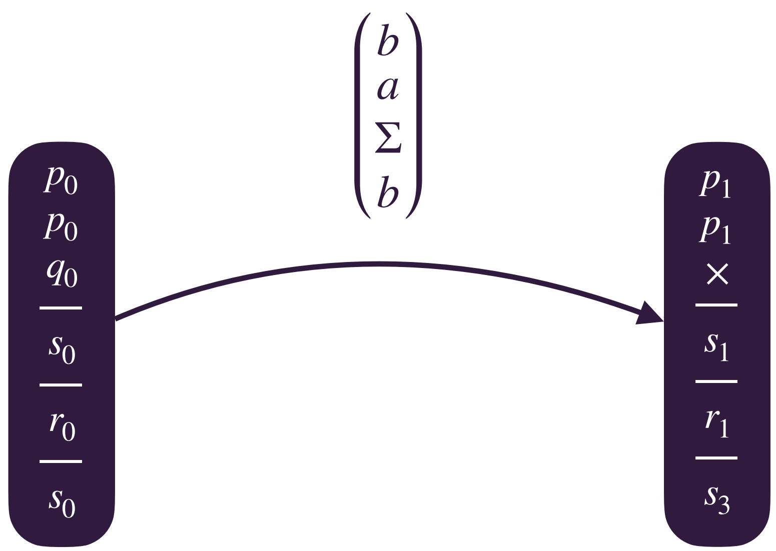


Construction of a finite automaton — 1

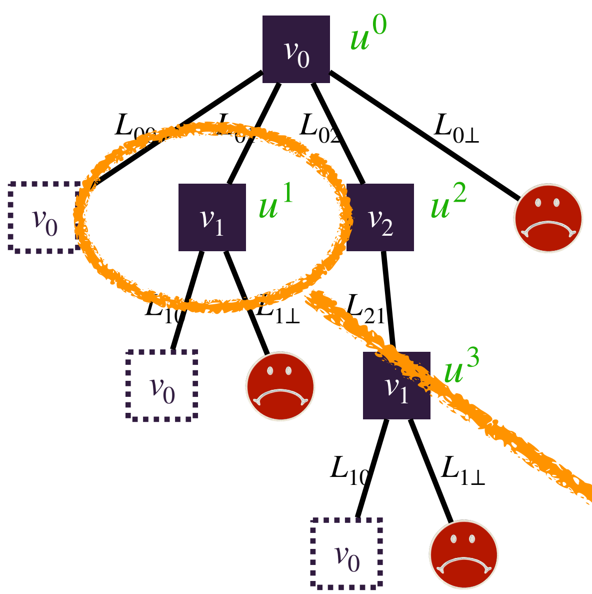




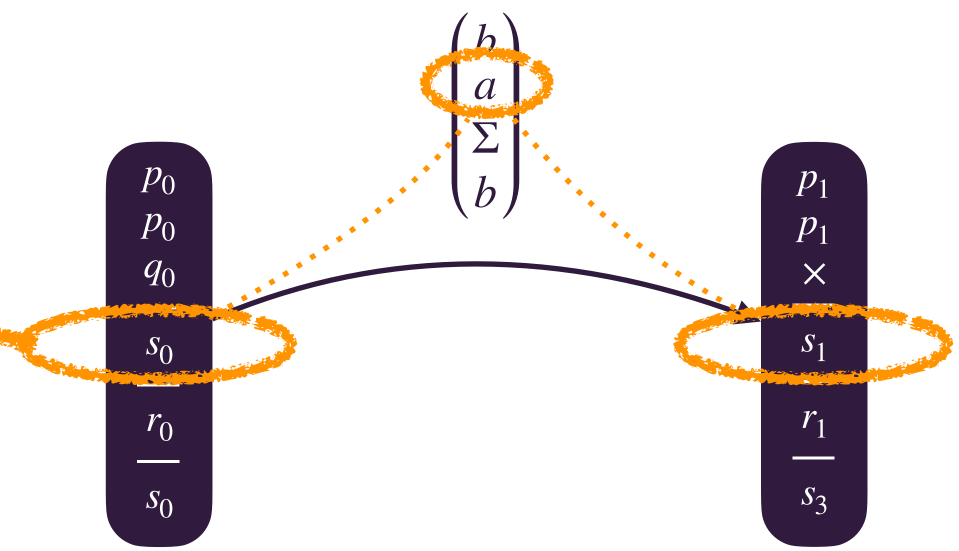
We launch several computations in parallel:

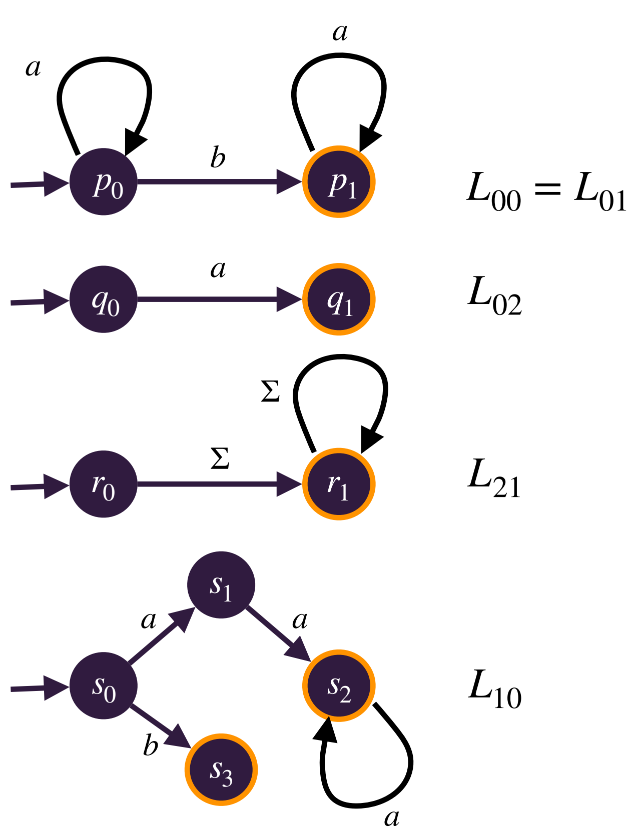


Construction of a finite automaton — 1

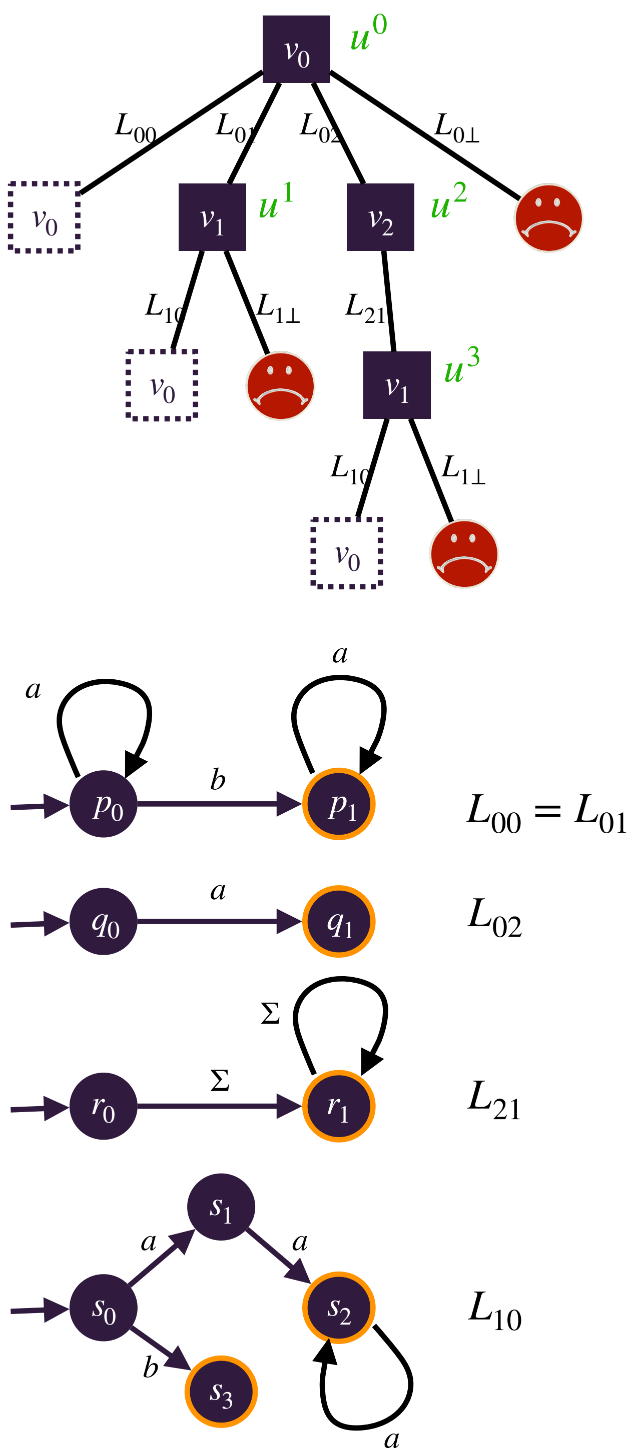


We launch several computations in parallel:

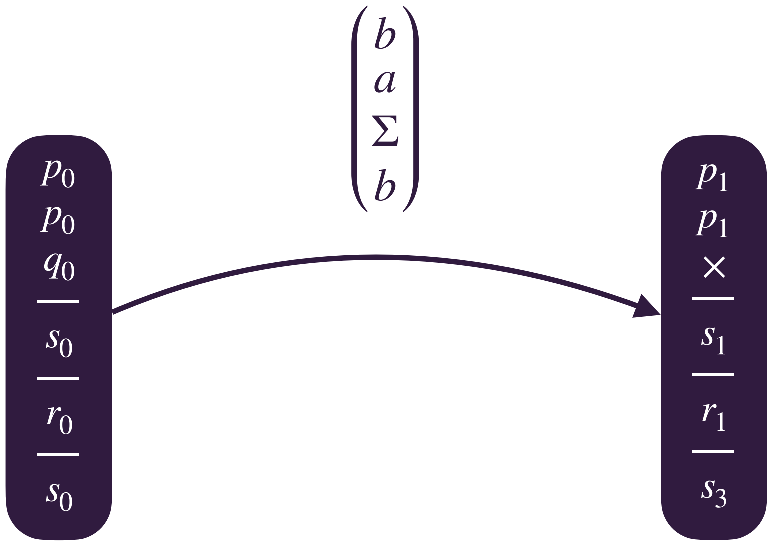




Construction of a finite automaton — 1

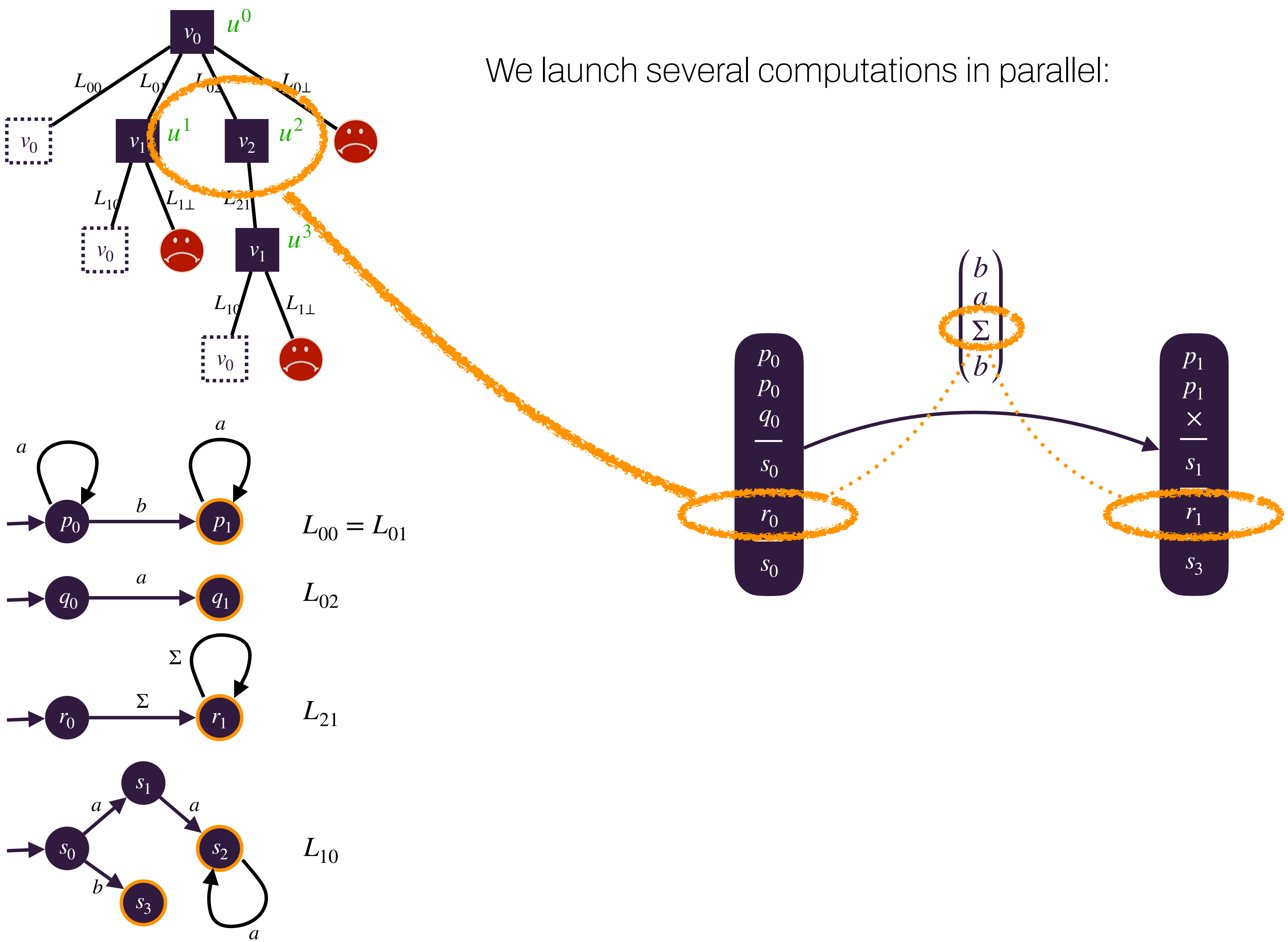


We launch several computations in parallel:

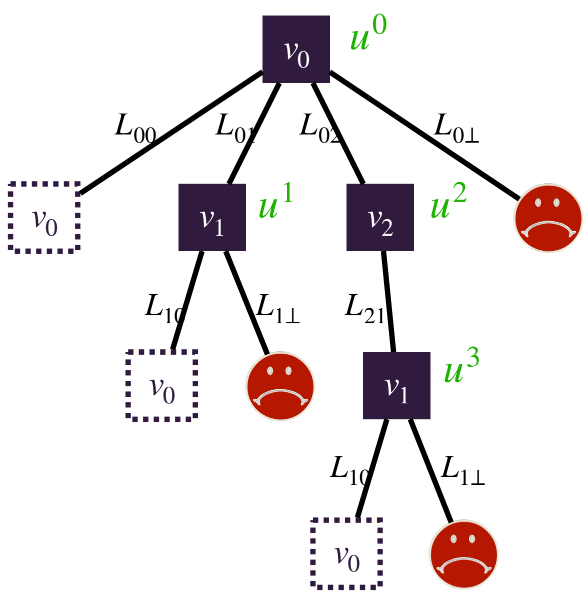


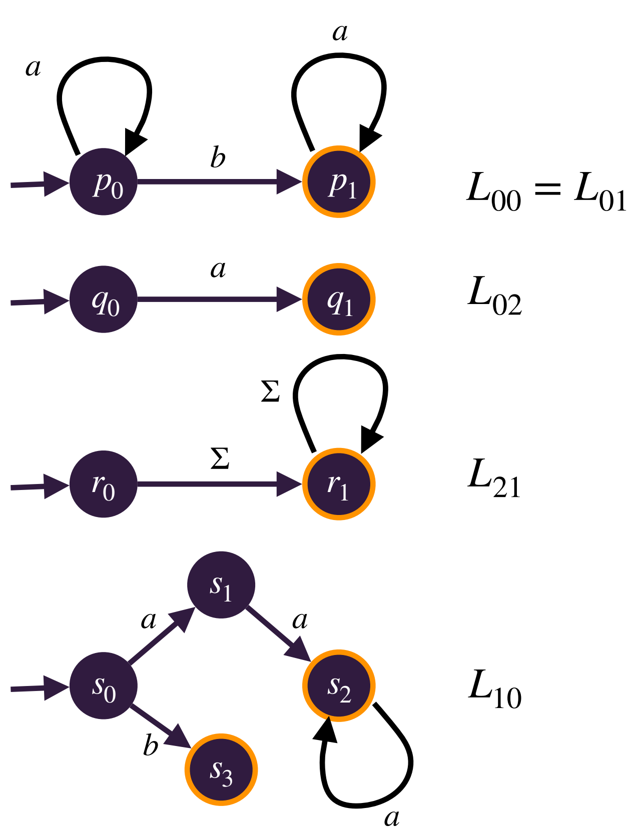
Construction of a finite automaton — 1

We launch several computations in parallel:

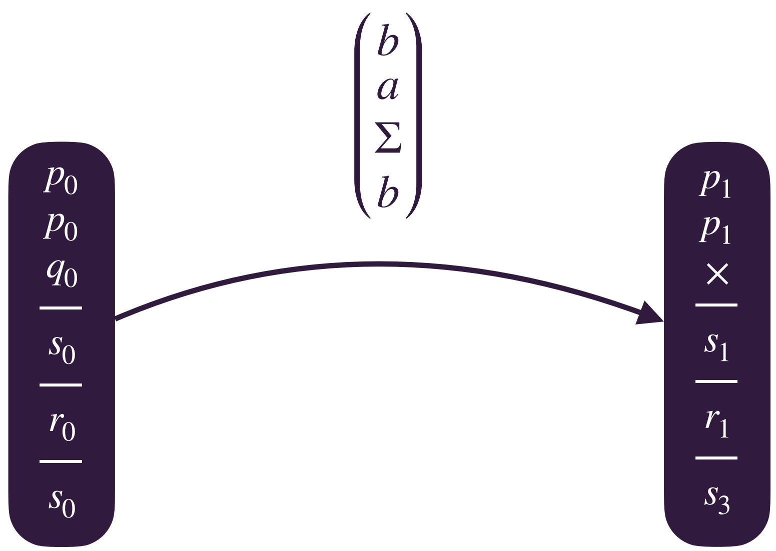


Construction of a finite automaton — 1



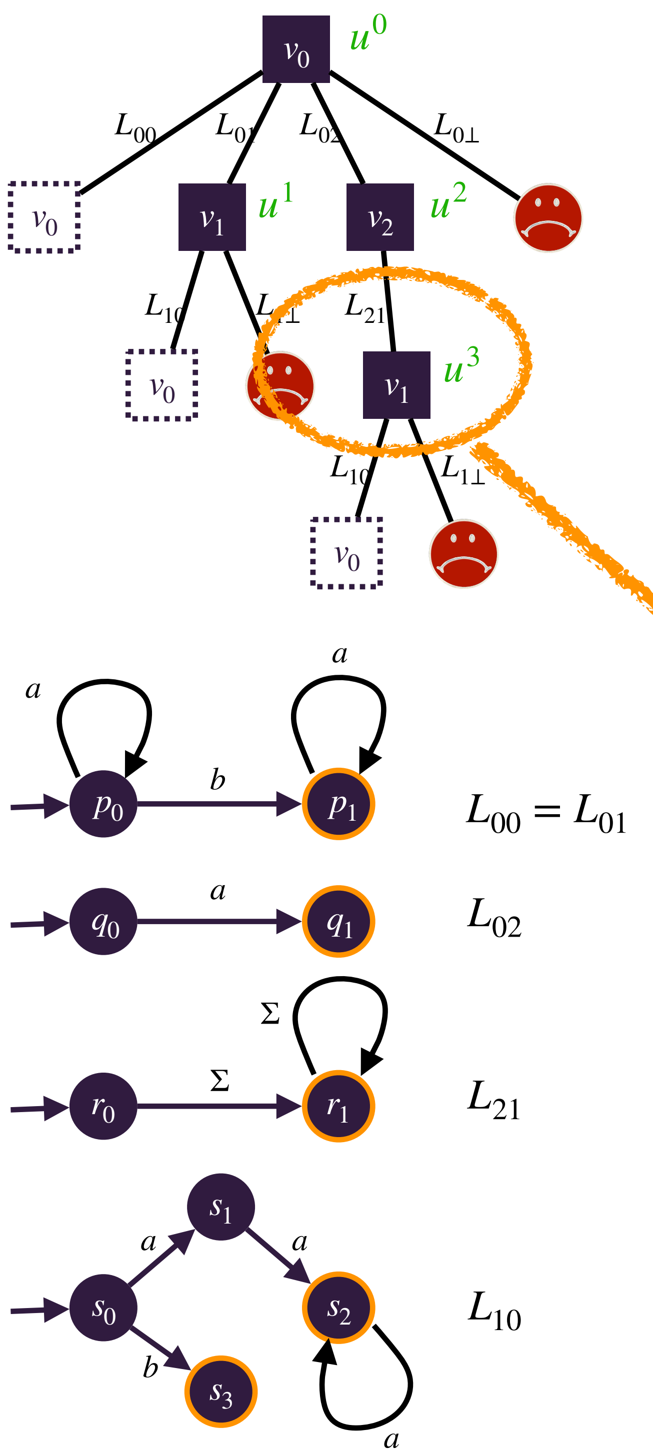


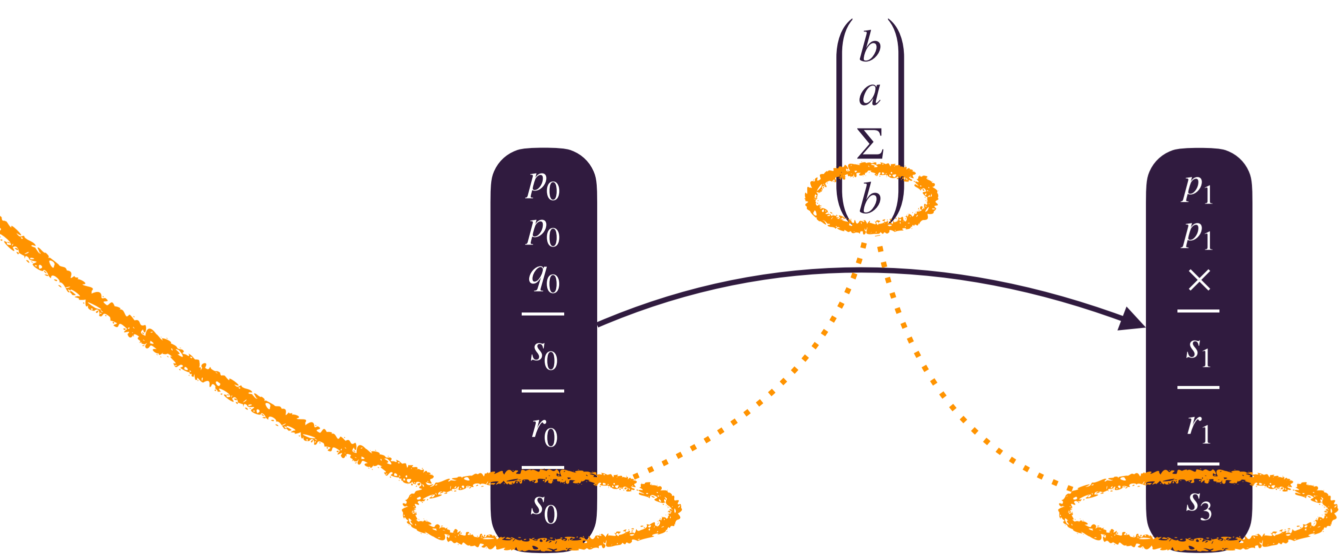
We launch several computations in parallel:



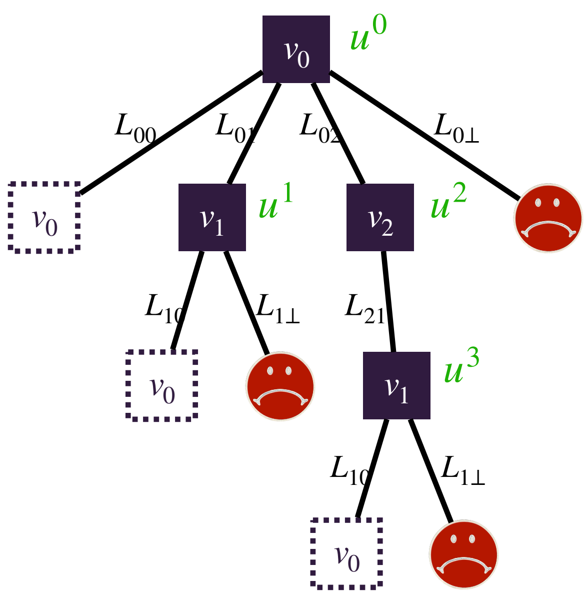
Construction of a finite automaton — 1

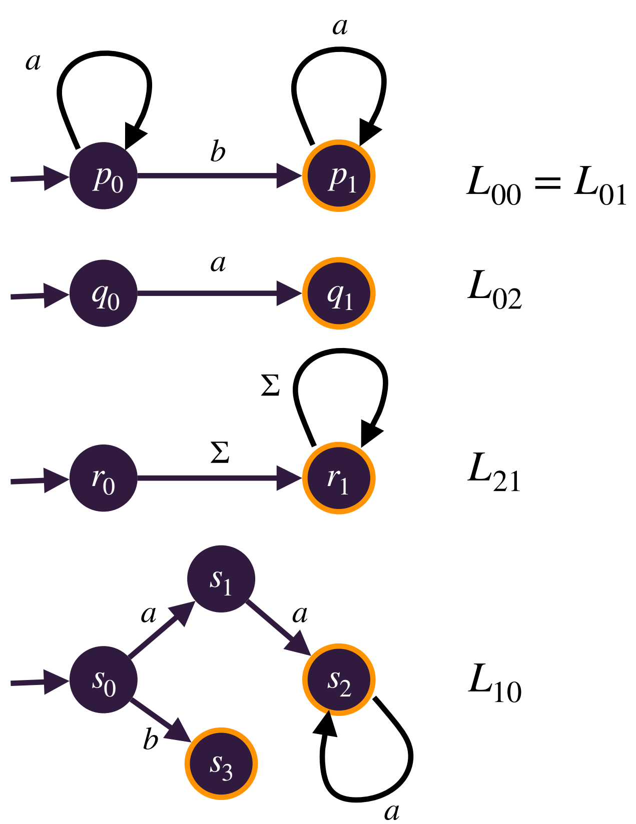
We launch several computations in parallel:



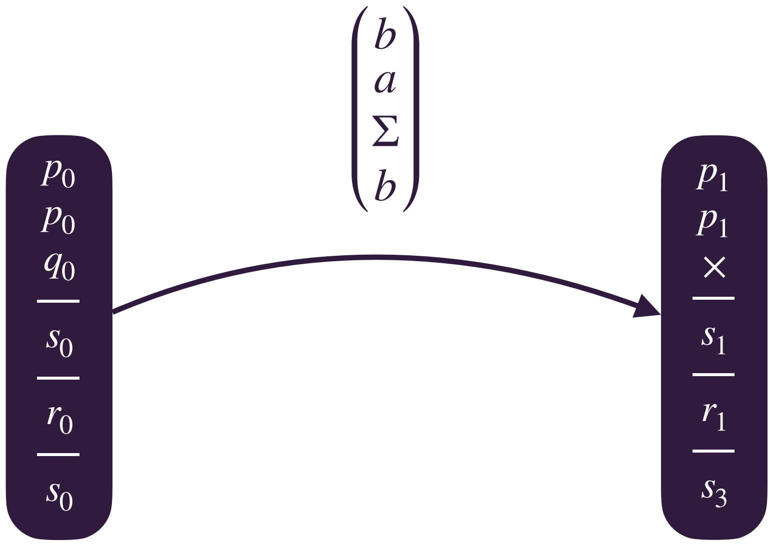


Construction of a finite automaton — 1

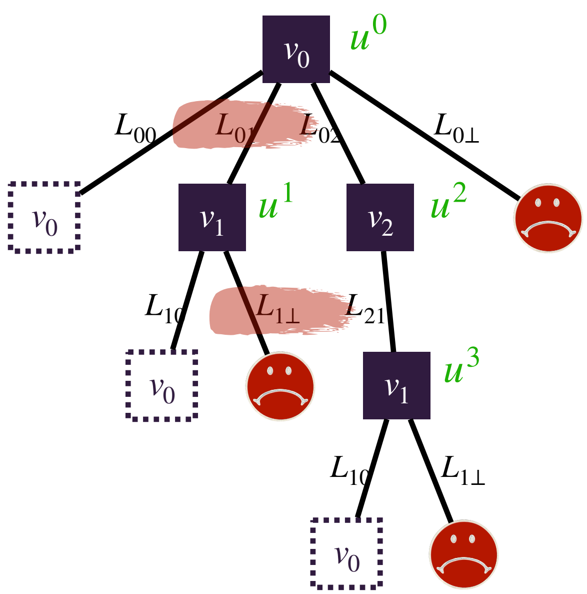


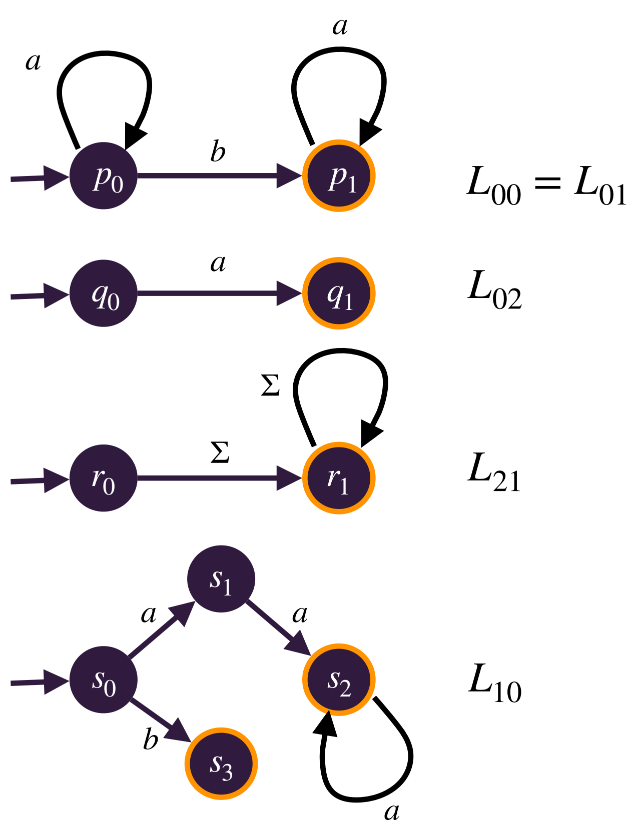


We launch several computations in parallel:

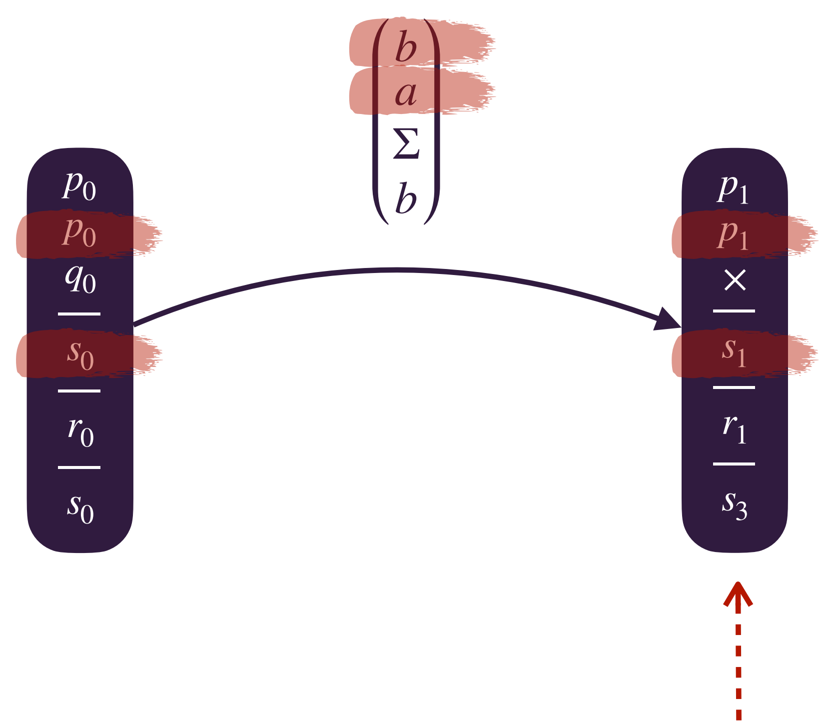


Construction of a finite automaton — 1





We launch several computations in parallel:

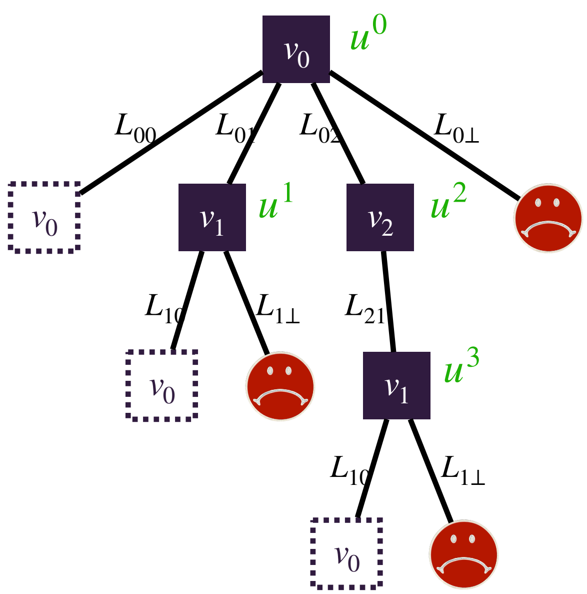


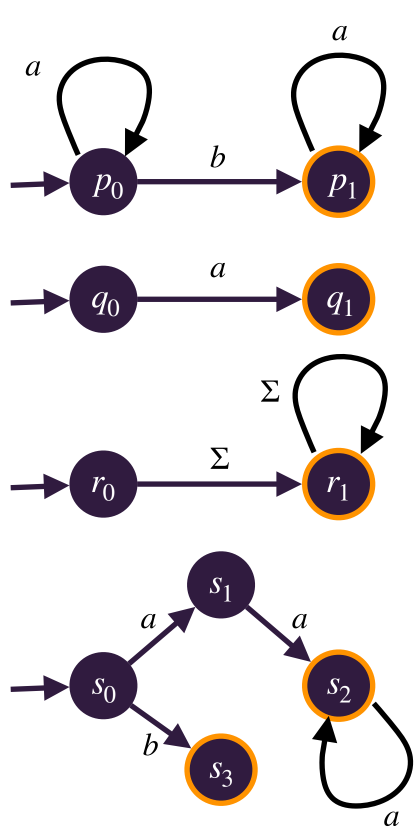
This state is unsafe, due to the branch $v_0 v_1$

$$b \in L_{0,1}, a \in L_{1\perp} = \Sigma^+ \setminus L_{10}$$

(p_1 accepting, s_1 not accepting)

Construction of a finite automaton — 2





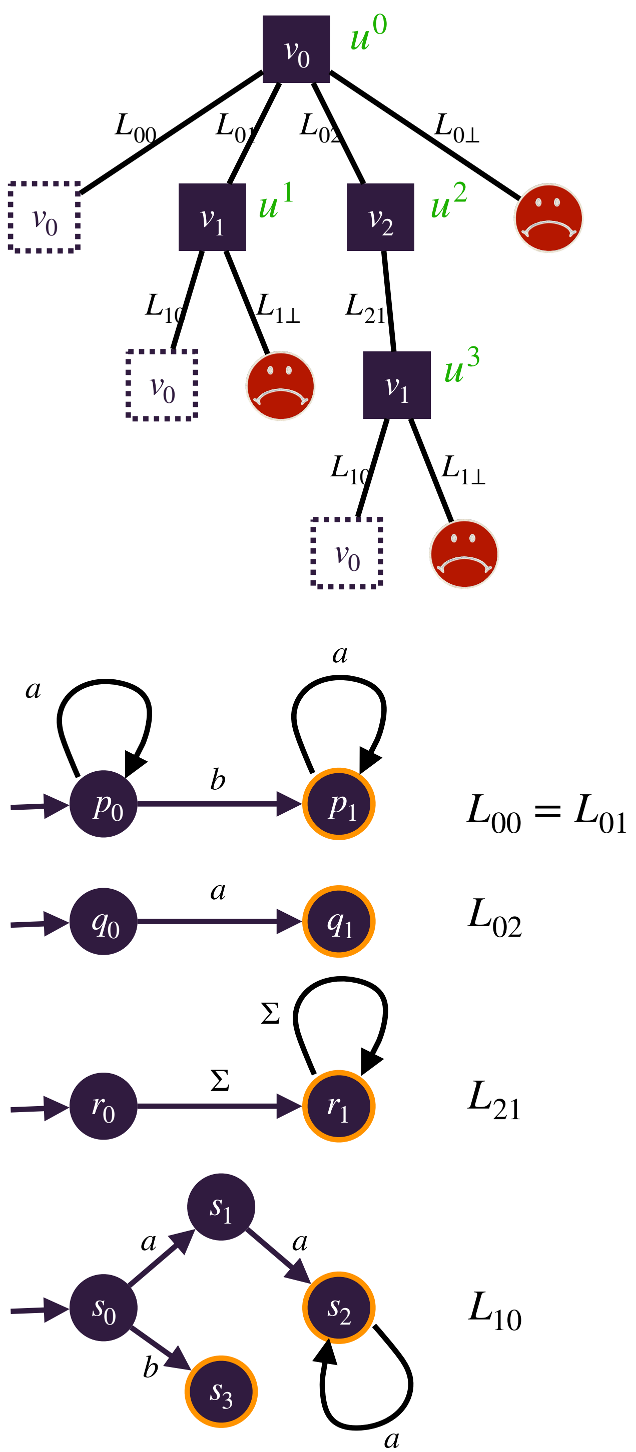
$L_{00} = L_{01}$

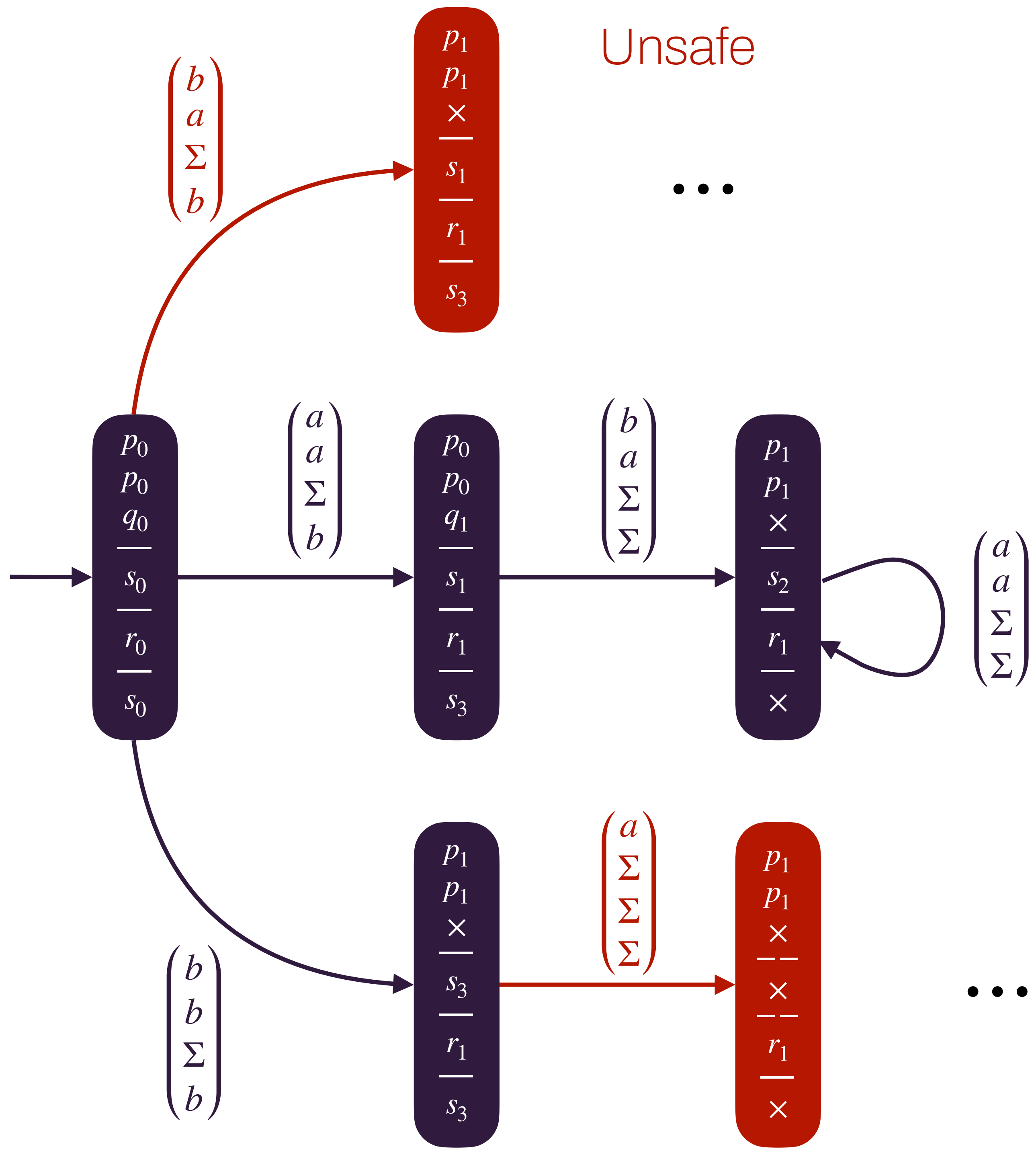
L_{02}

L_{21}

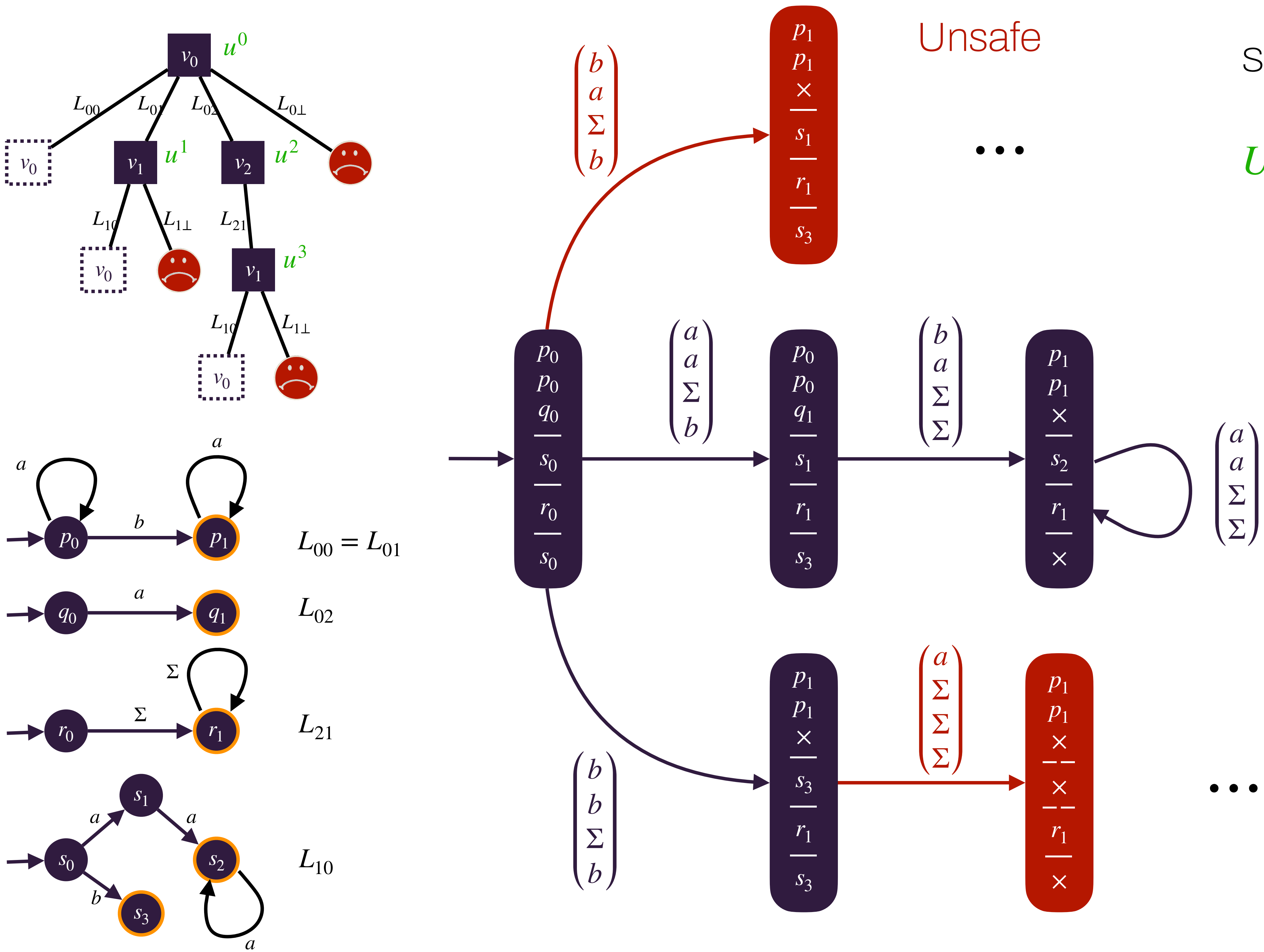
L_{10}

Construction of a finite automaton — 2





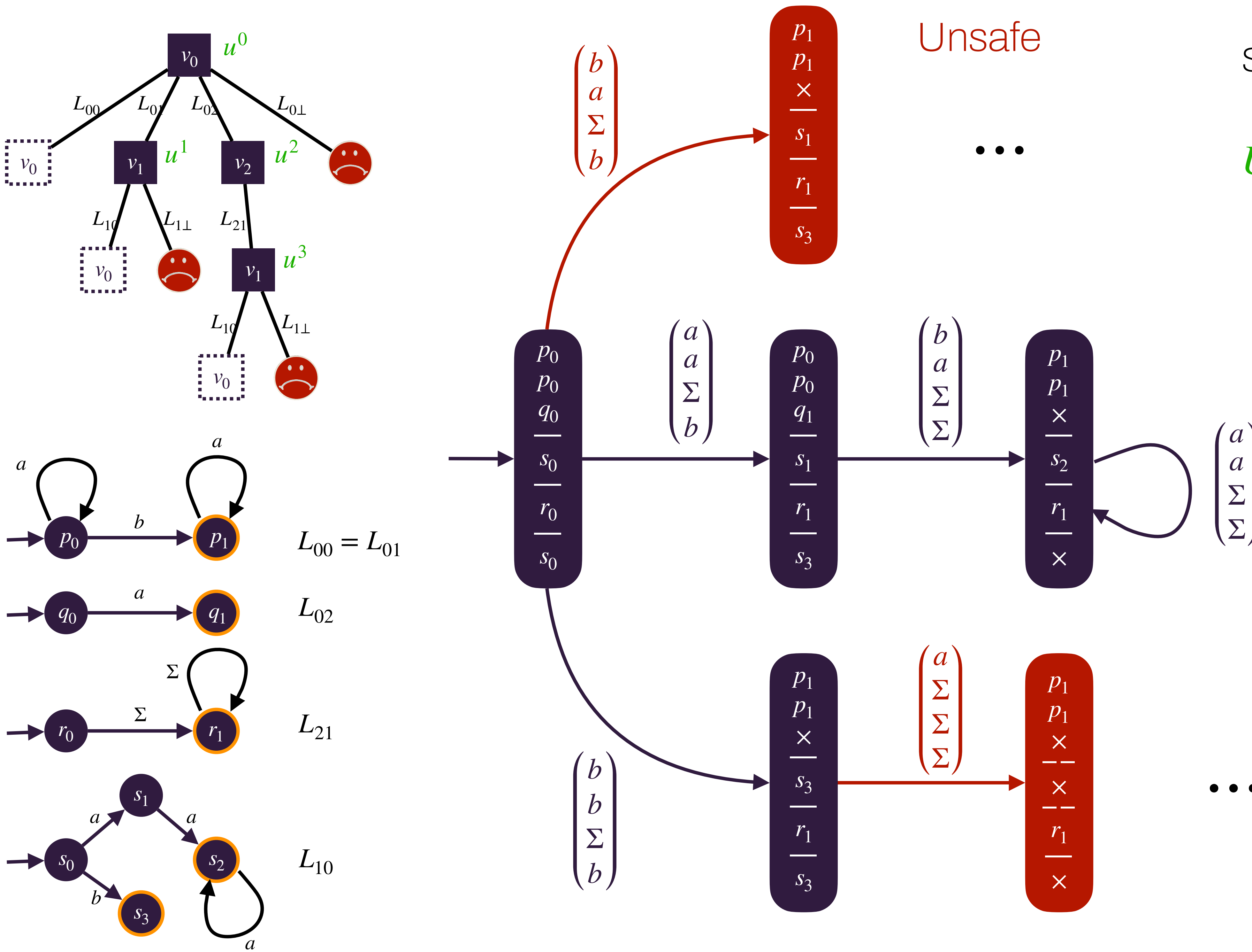
Construction of a finite automaton — 2



Safe word:

$$U = \begin{pmatrix} a \\ a \\ a \\ b \end{pmatrix} \begin{pmatrix} b \\ a \\ a \\ a \end{pmatrix} \begin{pmatrix} a \\ a \\ a \\ a \end{pmatrix}^{\omega}$$

Construction of a finite automaton — 2

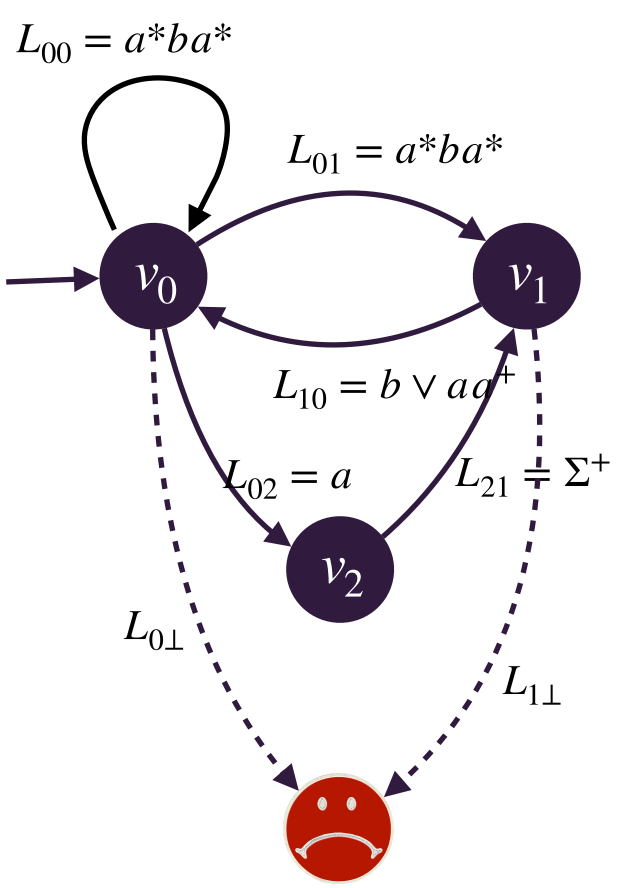


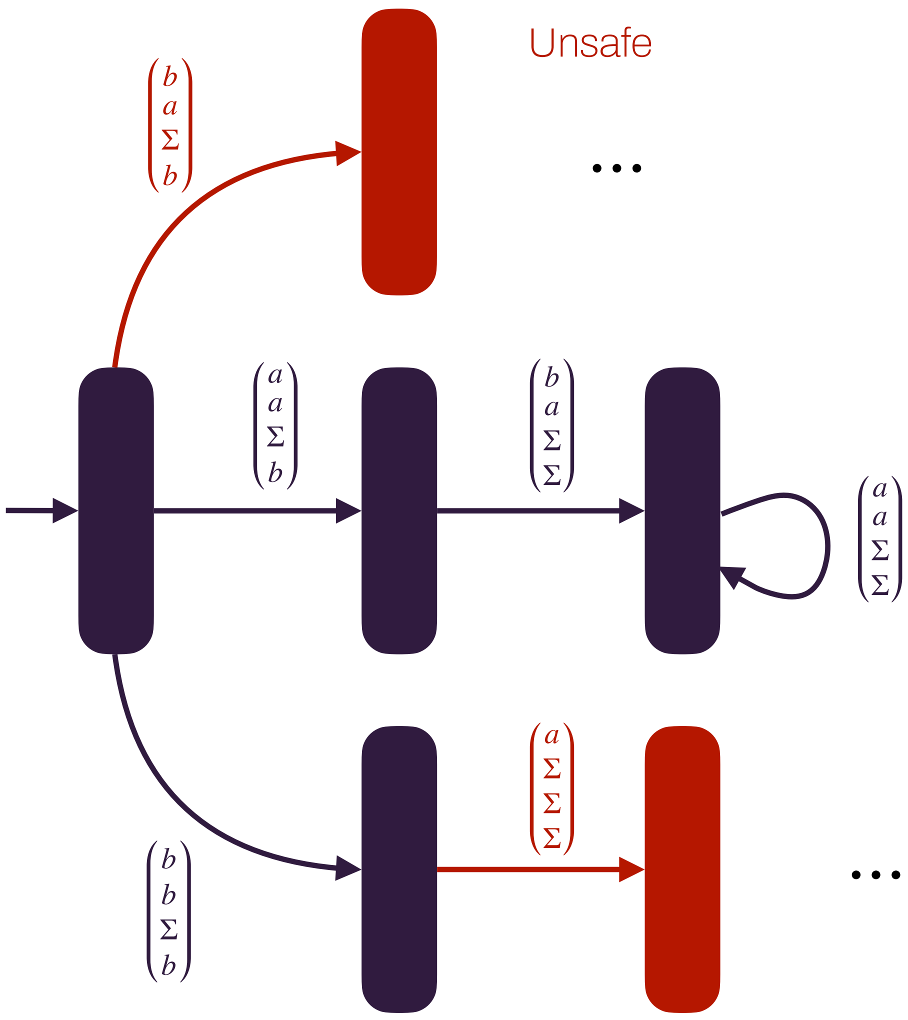
Safe word:

$$U = \begin{pmatrix} a \\ a \\ a \\ b \end{pmatrix} \begin{pmatrix} b \\ a \\ a \\ a \end{pmatrix} \begin{pmatrix} a \\ a \\ a \\ a \end{pmatrix}^{\omega}$$

$$u^0 = aba^{\omega}$$
$$u^1 = a^{\omega}$$
$$u^2 = a^{\omega}$$
$$u^3 = ba^{\omega}$$

Recap





There is a winning safe coalition strategy in the game iff there is an infinite safe word in the constructed automaton

The result

Decidability/complexity results

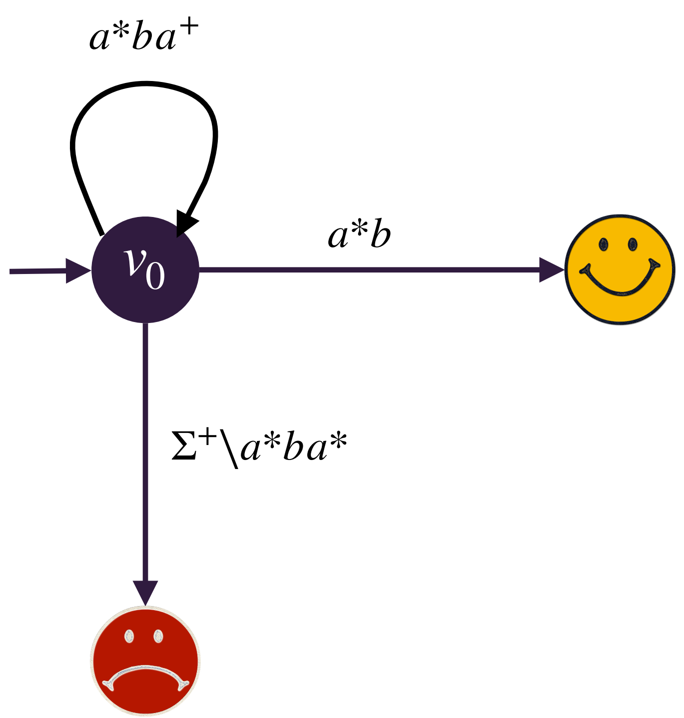
The safety coalition problem is decidable in EXPSPACE. It is PSPACE-hard.

- Upper bound: the size of the pruned unfolding can be exponential (and not possible to consider a polynomial-size DAG instead)
- Lower bound: similar reduction as for the strong controller synthesis from QSAT

Going further?

Going further?

- Understand the case of other objectives, starting with Reachability

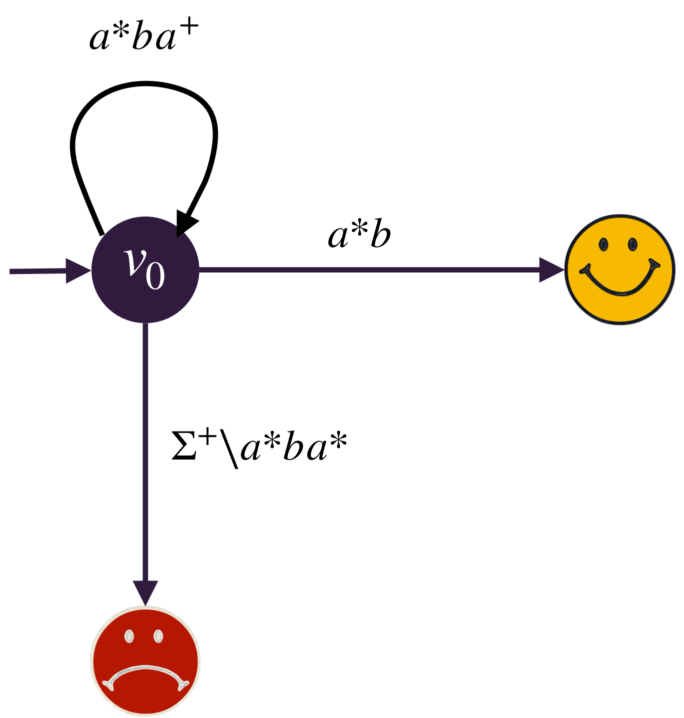


A winning coalition strategy

- At round i:
  - Player i plays b
  - Player $j \neq i$ plays a
- At round i, coalition plays $a^{i-1}ba^\omega$

Going further?

- Understand the case of other objectives, starting with Reachability



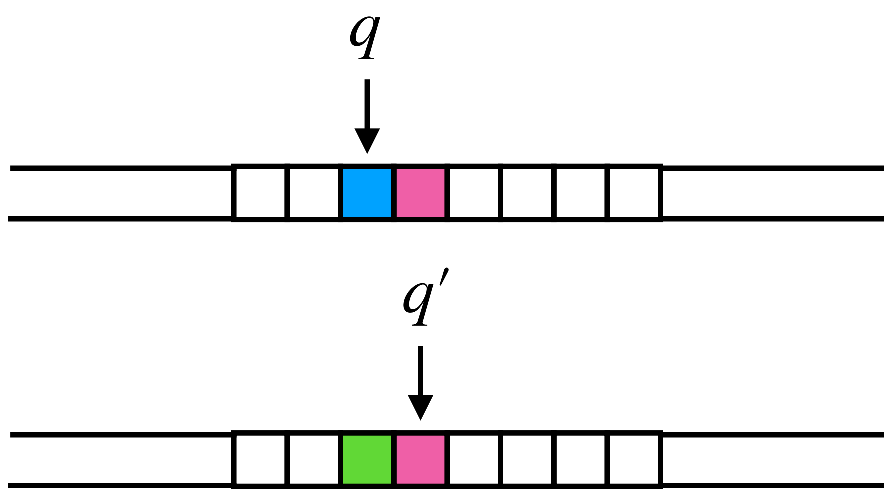
A winning coalition strategy

- At round i:
  - Player i plays b
  - Player $j \neq i$ plays a
- At round i, coalition plays $a^{i-1}ba^{\omega}$

- Limits: undecidability if regular relations instead of regular languages

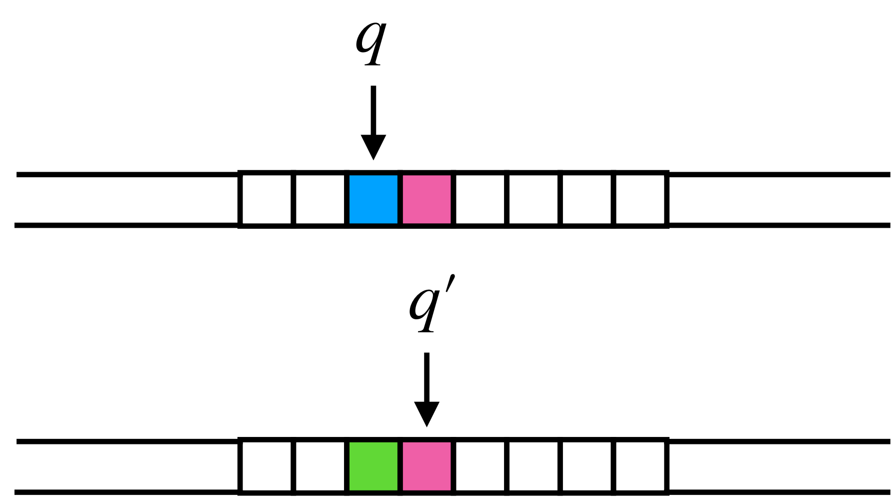
Undecidability under rational relations

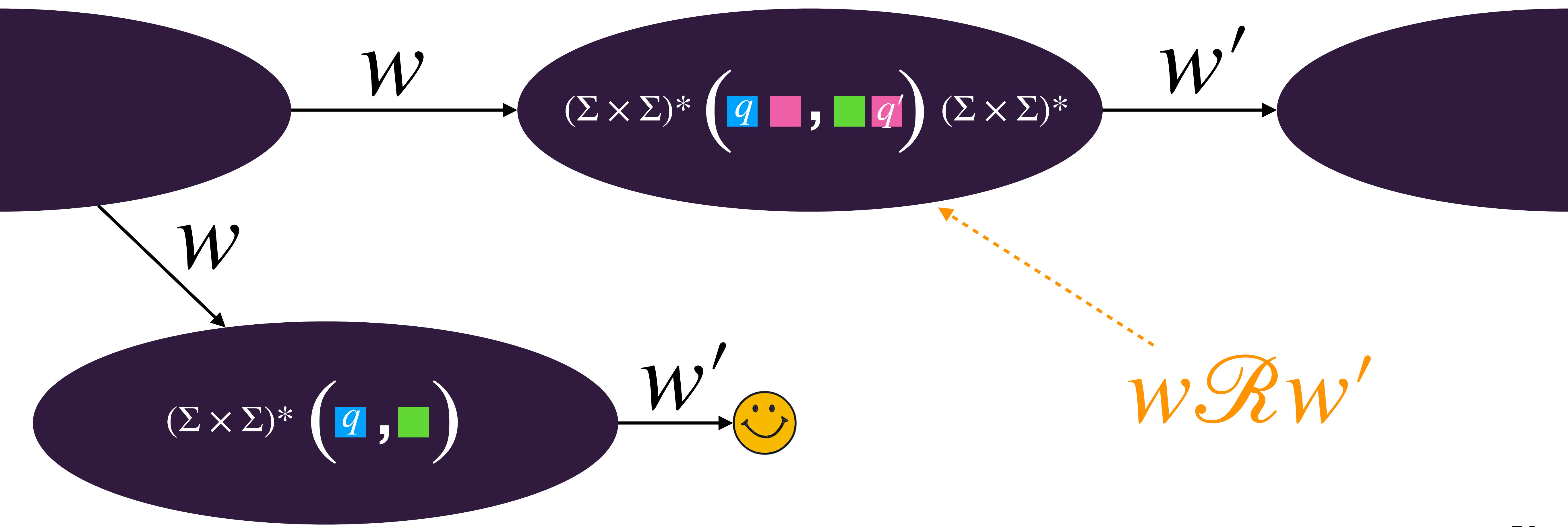
Det. Turing machine $\mathscr{M}$



Undecidability under rational relations

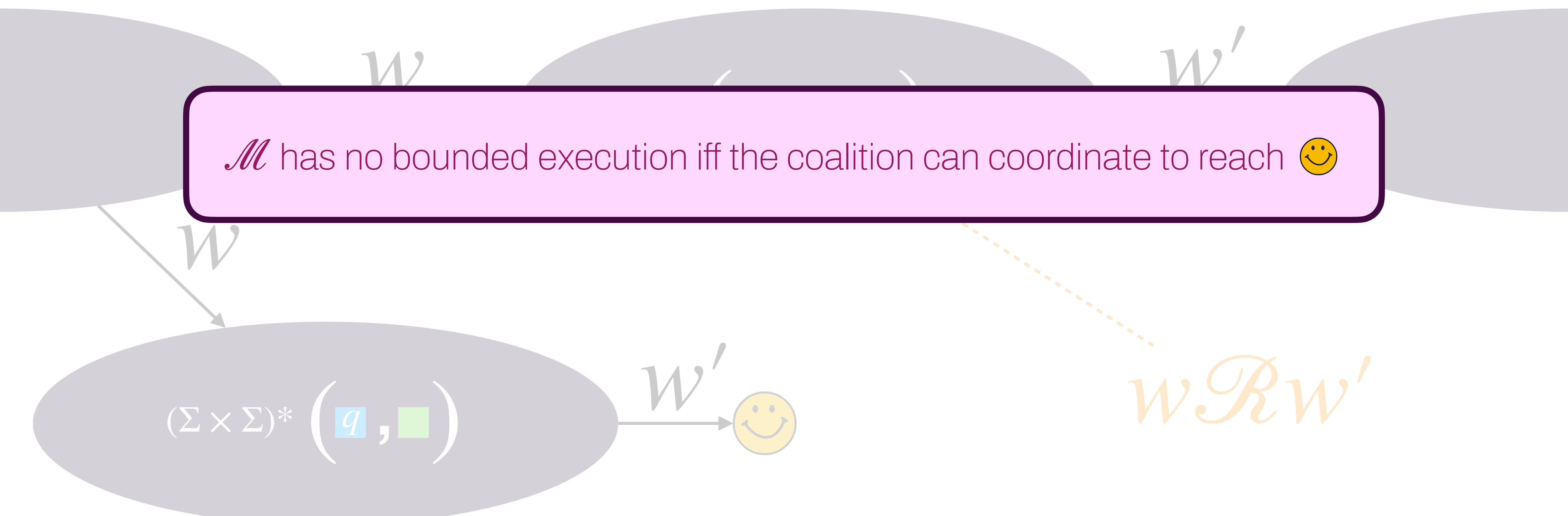
Det. Turing machine $\mathscr{M}$





Undecidability under rational relations

Det. Turing machine $\mathscr{M}$



$\mathscr{M}$ has no bounded execution iff the coalition can coordinate to reach 🙂

Conclusion and further work

Summary

Summary

- A concurrent parameterized game model
  - To reason about an unbounded number of agents
  - A natural extension of standard concurrent games

Summary

- A concurrent parameterized game model
  - To reason about an unbounded number of agents
  - A natural extension of standard concurrent games
- Two natural problems under inspection:
  - The crowd controller problem
  - The coalition problem

Further work

Further work

- Some technical further work:
  - Better understand the coalition problem

Further work

- Some technical further work:
  - Better understand the coalition problem
- Investigate solution concepts relevant to multiplayer games?
  - Various notions of rational behaviors (e.g. equilibria)

Further work

- Some technical further work:
  - Better understand the coalition problem
- Investigate solution concepts relevant to multiplayer games?
  - Various notions of rational behaviors (e.g. equilibria)
- Integrate new features in the model for better modeling power
  - Add partial information?
  - Infinite state space useful?
  - More general structures than words?

Questions?