

Antichains for Concurrent Parameterized Games

GandALF 2026

Nathalie Bertrand, Patricia Bouyer, Gaëtan Staquet

`gaetan.staquet@ec-nantes.fr`

`www.gaetanstaquet.com`

École Centrale de Nantes – LS2N, Nantes, France

15th September 2026

Contents

1. Concurrent parameterized games
2. A symbolic approach
 1. Antichains in a nutshell
 2. Fixed-point characterization
3. Experimental results

Parameterized systems

Print this!

Printer



Parameterized systems

Print this!

Printer



Print this!

Parameterized systems

Print this!

Printer



Print this!

Print this!

Parameterized systems

Print this!

Printer

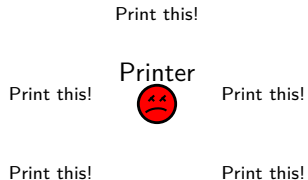


Print this!

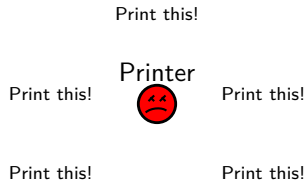
Print this!

Print this!

Parameterized systems



Parameterized systems



Parameterized systems:

- ▶ We do not know **how many agents** are in play.
- ▶ The systems must be correct **no matter how many agents** we have.

Constraints in this work

Deciding whether a parameterized system is correct is undecidable in general.¹

So, we restrict the models we consider. Here:

- ▶ Games on graphs.
- ▶ Focus on one agent (**Eve**) against a coalition of adversaries (**Adam**).
- ▶ The number of adversaries in Adam is **fixed, but unknown for Eve**.
- ▶ **Eve selects** an **action**. According to **the size of Adam**, **some transitions** are enabled. **Adam** (collectively) **chooses** which **transition** to follow.

↪ **Concurrent parameterized games**.²

¹Apt and Kozen, “Limits for automatic verification of finite-state concurrent systems”, 1986.

²Bertrand, Bouyer, and Majumdar, “Concurrent Parameterized Games”, 2019.

Concurrent parameterized games (on graphs)

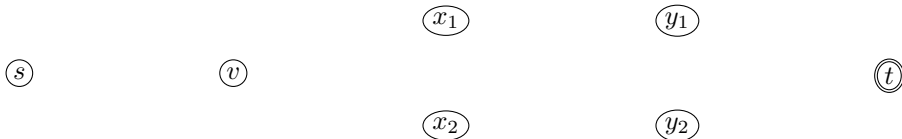
Definition 1 (Concurrent parameterized games). A **concurrent parameterized game** is a tuple $\mathcal{A} = (\Sigma, V, \Delta)$ where

- ▶ Σ is a finite, non-empty set of **actions**;

Concurrent parameterized games (on graphs)

Definition 1 (Concurrent parameterized games). A **concurrent parameterized game** is a tuple $\mathcal{A} = (\Sigma, V, \Delta)$ where

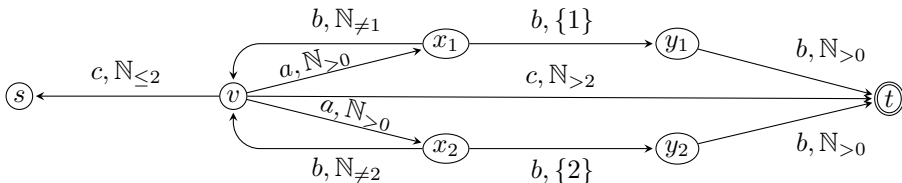
- ▶ Σ is a finite, non-empty set of **actions**;
- ▶ V is a finite, non-empty set of **vertices**; and



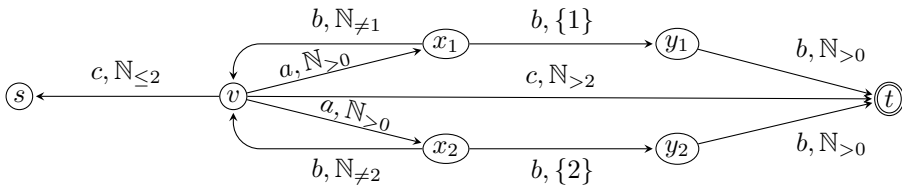
Concurrent parameterized games (on graphs)

Definition 1 (Concurrent parameterized games). A **concurrent parameterized game** is a tuple $\mathcal{A} = (\Sigma, V, \Delta)$ where

- ▶ Σ is a finite, non-empty set of **actions**;
- ▶ V is a finite, non-empty set of **vertices**; and
- ▶ $\Delta : V \times \Sigma \times \mathbb{N}_{>0} \rightarrow \mathcal{P}(V)$ is the **transition function**.



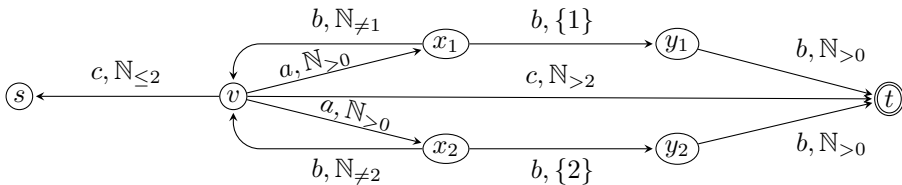
Concurrent parameterized games – Into the unknowns



If Eve plays b from x_1 :

- ▶ Exactly one opponent? Go to y_1 .
- ▶ Otherwise, go to v .

Concurrent parameterized games – Into the unknowns

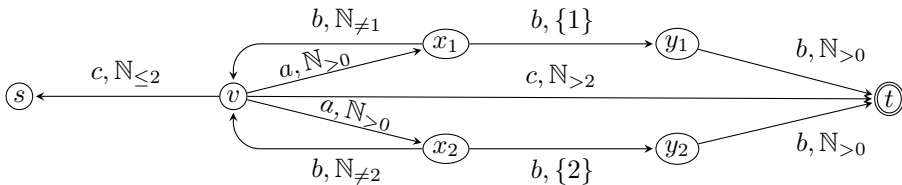


If Eve plays b from x_1 :

- ▶ Exactly one opponent? Go to y_1 .
- ▶ Otherwise, go to v .

If Eve plays a from v : Adam picks x_1 or x_2 .

Concurrent parameterized games – Into the unknowns



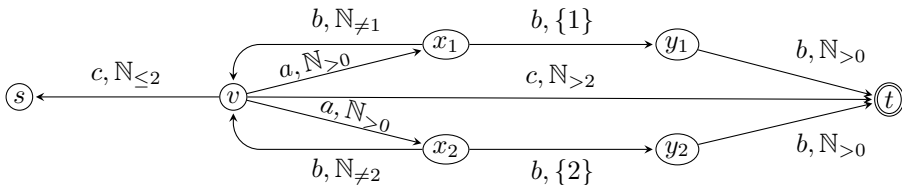
If Eve plays b from x_1 :

- ▶ Exactly one opponent? Go to y_1 .
- ▶ Otherwise, go to v .

If Eve plays a from v : Adam picks x_1 or x_2 .

↪ Two sources of unknown: the **non-determinism** and the **number of opponents** k .

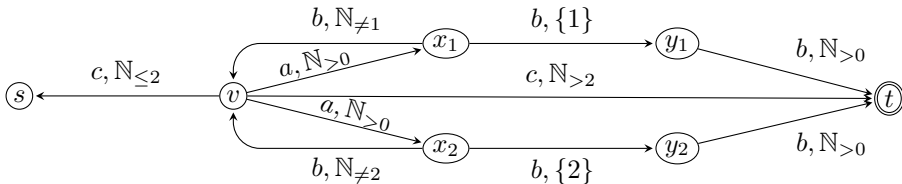
Concurrent parameterized games – Enabled actions



Arena assumptions. If an action is enabled for some number of opponents, it must be enabled for every number of opponents.

The arena has a **finite representation**: here, constraints on the edges are given as **finite unions of intervals**.

Concurrent parameterized games – Enabled actions



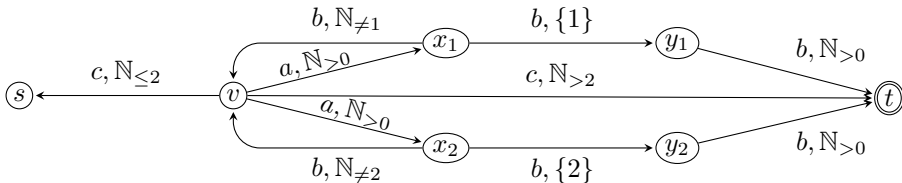
Arena assumptions. If an action is enabled for some number of opponents, it must be enabled for every number of opponents.

The arena has a **finite representation**: here, constraints on the edges are given as **finite unions of intervals**.

Δ gives us where to go when playing against a number of opponents, e.g., $\Delta(x_1, b, \mathbb{N}_{\neq 1}) = \{v\}$.

To know the number of opponents to go from v to v' : $\nabla(v, a, v')$, e.g., $\nabla(x_1, b, v) = \mathbb{N}_{\neq 1}$.

Reachability objective



Objectives.

- ▶ Eve: **reach** t .
- ▶ Adam: prevent her from reaching t .

Strategies

Definition 2 (k -history). **Fix** $k \in \mathbb{N}_{>0}$. A **k -history** is a finite sequence $v_0 a_0 \cdots v_i$ in $(V \cdot \Sigma)^* \cdot V$ such that for every j , $v_{j+1} \in \Delta(v_j, a_j, k)$.
A **history** is a k -history for some $k \in \mathbb{N}_{>0}$.
We write Hist for the set of all histories.

Strategies

Definition 2 (k -history). **Fix** $k \in \mathbb{N}_{>0}$. A **k -history** is a finite sequence $v_0 a_0 \cdots v_i$ in $(V \cdot \Sigma)^* \cdot V$ such that for every j , $v_{j+1} \in \Delta(v_j, a_j, k)$.
A **history** is a k -history for some $k \in \mathbb{N}_{>0}$.
We write Hist for the set of all histories.

Definition 3 (Strategy). A **strategy** is a function $\sigma : \text{Hist} \rightarrow \Sigma$.

A strategy is **winning** from a vertex v if it **guarantees** to reach t from v , **no matter the number of opponents**.

Strategies

Definition 2 (k -history). **Fix** $k \in \mathbb{N}_{>0}$. A **k -history** is a finite sequence $v_0 a_0 \cdots v_i$ in $(V \cdot \Sigma)^* \cdot V$ such that for every j , $v_{j+1} \in \Delta(v_j, a_j, k)$.
A **history** is a k -history for some $k \in \mathbb{N}_{>0}$.
We write Hist for the set of all histories.

Definition 3 (Strategy). A **strategy** is a function $\sigma : \text{Hist} \rightarrow \Sigma$.

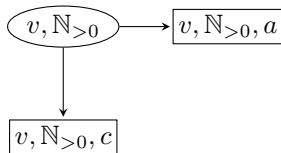
A strategy is **winning** from a vertex v if it **guarantees** to reach t from v , **no matter the number of opponents**.

THE question. What is the set of vertices from which Eve has a winning strategy? In other words: what is her **winning region**?

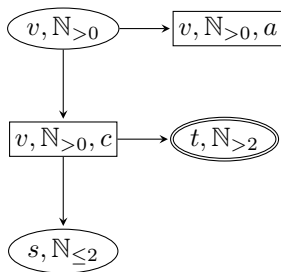
Knowledge game: a standard two-player turn-based game

$$v, \mathbb{N}_{>0}$$

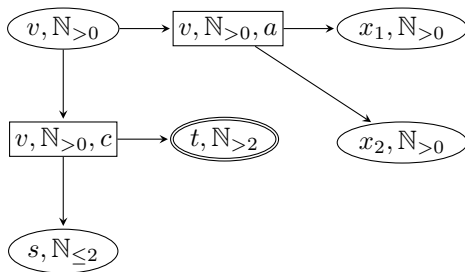
Knowledge game: a standard two-player turn-based game



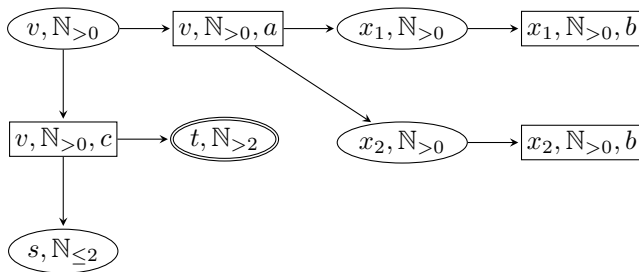
Knowledge game: a standard two-player turn-based game



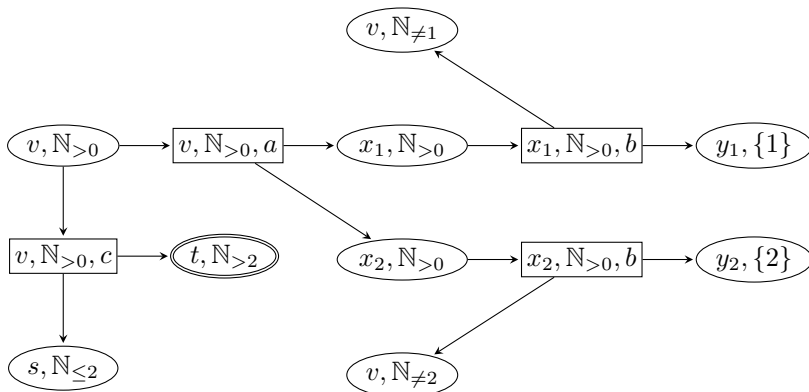
Knowledge game: a standard two-player turn-based game



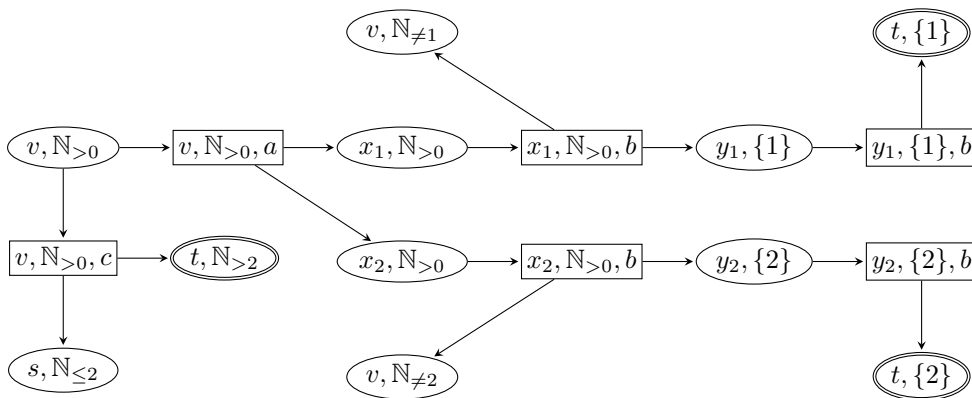
Knowledge game: a standard two-player turn-based game



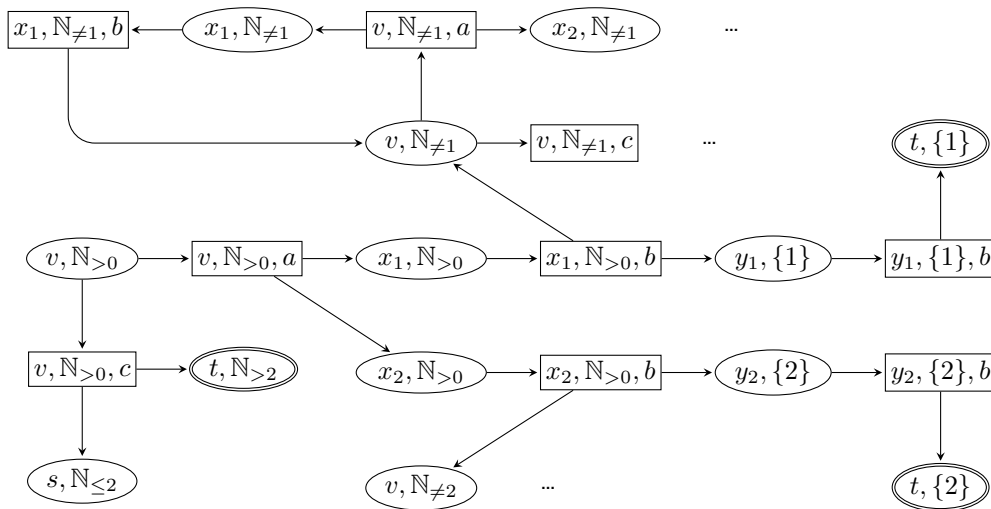
Knowledge game: a standard two-player turn-based game



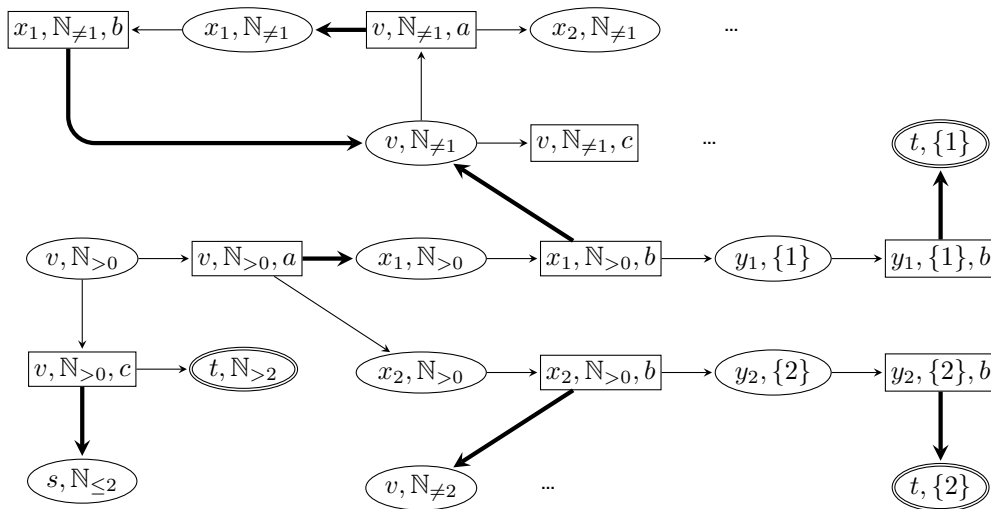
Knowledge game: a standard two-player turn-based game



Knowledge game: a standard two-player turn-based game



Knowledge game: a standard two-player turn-based game



Known results and new goals

Theorem 4 (Bertrand, Bouyer, and Majumdar, “Concurrent Parameterized Games”, 2019).

- ▶ The **knowledge game** is finite, and has a size **exponential** in the size of the **original game**.
- ▶ Eve has a winning strategy from v **in the original game**
if and only if
she has a winning strategy from $(v, \mathbb{N}_{>0})$ **in the knowledge game**.
- ▶ Deciding the existence of a winning strategy for Eve is PSPACE-complete.

Intuition for the PSPACE algorithm: construct the knowledge game on the fly, and explore it in a (clever) DFS way.

Known results and new goals

Theorem 4 (Bertrand, Bouyer, and Majumdar, “Concurrent Parameterized Games”, 2019).

- ▶ The **knowledge game** is finite, and has a size **exponential** in the size of the **original game**.
- ▶ Eve has a winning strategy from v **in the original game**
if and only if
she has a winning strategy from $(v, \mathbb{N}_{>0})$ **in the knowledge game**.
- ▶ Deciding the existence of a winning strategy for Eve is PSPACE-complete.

Intuition for the PSPACE algorithm: construct the knowledge game on the fly, and explore it in a (clever) DFS way.

Limitation: can only decide whether Eve can win from **a fixed v** .

Known results and new goals

Theorem 4 (Bertrand, Bouyer, and Majumdar, “Concurrent Parameterized Games”, 2019).

- ▶ The **knowledge game** is finite, and has a size **exponential** in the size of the **original game**.
- ▶ Eve has a winning strategy from v **in the original game**
if and only if
she has a winning strategy from $(v, \mathbb{N}_{>0})$ **in the knowledge game**.
- ▶ Deciding the existence of a winning strategy for Eve is PSPACE-complete.

Intuition for the PSPACE algorithm: construct the knowledge game on the fly, and explore it in a (clever) DFS way.

Limitation: can only decide whether Eve can win from **a fixed v** .

Research question. How to efficiently compute the **whole winning region** of Eve?

Some observations

Suppose Eve can win from some v against

- ▶ exactly one opponent ($\{1\}$); and
- ▶ exactly two opponents ($\{2\}$); and
- ▶ either one or two opponents ($\{1, 2\}$).

Then, we only need to remember $\{1, 2\}$.

Some observations

Suppose Eve can win from some v against

- ▶ exactly one opponent ($\{1\}$); and
- ▶ exactly two opponents ($\{2\}$); and
- ▶ either one or two opponents ($\{1, 2\}$).

Then, we only need to remember $\{1, 2\}$.

We only need the **maximal elements** (according to some order).

Antichains!

Partially ordered sets and dominated elements

Let X be any set, and $L \subseteq X$.

▶ A **partial order** on X is a binary relation $\sqsubseteq \subseteq X \times X$ that is

- ▶ reflexive ($x \sqsubseteq x$);
- ▶ transitive ($x \sqsubseteq y \wedge y \sqsubseteq z \Rightarrow x \sqsubseteq z$); and
- ▶ antisymmetric ($x \sqsubseteq y \wedge y \sqsubseteq x \Rightarrow x = y$).

▶ Then, $(X, \sqsubseteq)$ is a **partially ordered set**.

▶ L is an **antichain** for $(X, \sqsubseteq)$ if, for any $x \neq x' \in L$, x and x' are **incomparable**:

$$x \not\sqsubseteq x' \wedge x' \not\sqsubseteq x.$$

▶ $\subseteq$ is a partial order on $\mathcal{P}(\mathbb{N}_{>0})$.

▶ $(\mathcal{P}(\mathbb{N}_{>0}), \subseteq)$ is a partially ordered set.

▶ $\{\{1, 2\}, \{5\}\}$ is an antichain for $(\mathcal{P}(\mathbb{N}_{>0}), \subseteq)$.

Antichains

► We say that an element $x \in X$ is **dominated** by L if we can find an element of L that subsumes x .

► The set of all elements dominated by L is:

$$\text{Dom}(L) = \{x \in X \mid \exists x' \in L : x \sqsubseteq x' \wedge x \neq x'\}.$$

► The **reduced form** of L is obtained by **removing** all the dominated elements:

$$\lceil L \rceil = L \setminus \text{Dom}(L).$$

► $\{2\}$ is dominated by $\{\{1, 2\}, \{5\}\}$.

► $\text{Dom}(\{\{1, 2\}, \{5\}\}) = \{\emptyset, \{1\}, \{2\}\}$.

► $\lceil \{\{1, 2\}, \{1\}, \{5\}\} \rceil = \{\{1, 2\}, \{5\}\}$.

Lemma 5. $\lceil L \rceil$ is an antichain (*i.e.*, a set of incomparable elements) for $(X, \sqsubseteq)$.

The maximal elements suffice (under some conditions)!

Back to the game again

Definition 6. We say that $(v, K) \sqsubseteq (v', K')$ if

- ▶ $v = v'$; and
- ▶ $K \subseteq K'$.

Then, $\sqsubseteq$ is a partial order.

Example 7.

- ▶ $(x_1, \mathbb{N}_{>0}) \not\sqsubseteq (x_2, \mathbb{N}_{>0})$, as $x_1 \neq x_2$.
- ▶ $(x_1, \{1, 2\}) \not\sqsubseteq (x_1, \{2, 3\})$, as $\{1, 2\} \not\subseteq \{2, 3\}$.
- ▶ $(x_1, \{1, 2\}) \sqsubseteq (x_1, \{1, 2, 3\})$.

Fixed-point characterizations – Outline

1. Eve can win from $(t, \mathbb{N}_{>0})$.
2. Compute the controllable predecessors of $(t, \mathbb{N}_{>0})$ to obtain a part of the winning region.
3. Keep only the maximal elements: **antichain** for the **known winning region**.
4. Repeat by computing the controllable predecessors of the known winning region.

Fixed-point characterizations – Outline

1. Eve can win from $(t, \mathbb{N}_{>0})$.
2. Compute the controllable predecessors of $(t, \mathbb{N}_{>0})$ to obtain a part of the winning region.
3. Keep only the maximal elements: **antichain** for the **known winning region**.
4. Repeat by computing the controllable predecessors of the known winning region.

How to compute the controllable predecessors? Two ways:

- ▶ W : Iterate over $\mathcal{P}(\mathbb{N}_{>0})$ and test whether the current set is “nice”.
- ▶ W_{Alt} : Construct nice sets from the constraints of the arena and the known winning region.

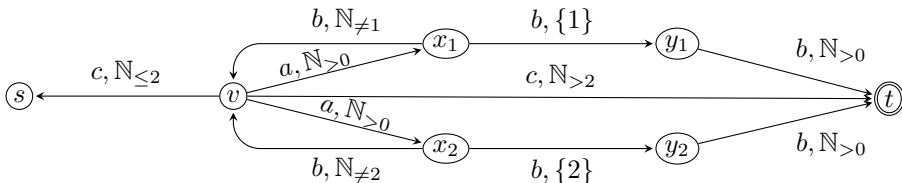
Fixed-point characterizations – Outline

1. Eve can win from $(t, \mathbb{N}_{>0})$.
2. Compute the controllable predecessors of $(t, \mathbb{N}_{>0})$ to obtain a part of the winning region.
3. Keep only the maximal elements: **antichain** for the **known winning region**.
4. Repeat by computing the controllable predecessors of the known winning region.

How to compute the controllable predecessors? Two ways:

- ▶ W : Iterate over $\mathcal{P}(\mathbb{N}_{>0})$ and test whether the current set is “nice”.
- ▶ W_{Alt} : Construct nice sets from the constraints of the arena and the known winning region.

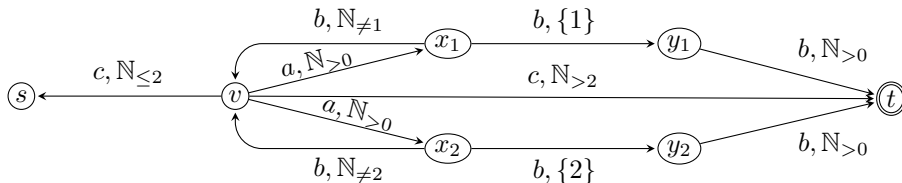
We will **never** construct the knowledge game via our fixed-point characterization.

W_{Alt} by example – First iteration: playing c from v 

We can win from s when playing
against this number of opponents

Starting with $W_{\text{Alt}}^0 = \{(\overbrace{s, \emptyset}, (v, \emptyset), (x_1, \emptyset), (x_2, \emptyset), (y_1, \emptyset), (y_2, \emptyset), (t, N_{>0}))\}$.

W_{Alt} by example – First iteration: playing c from v

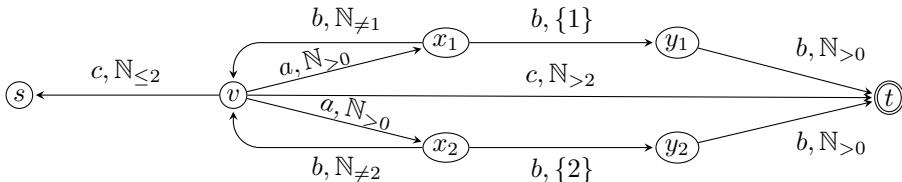


We can win from s when playing
against this number of opponents

Starting with $W_{\text{Alt}}^0 = \{\overbrace{(s, \emptyset)}, (v, \emptyset), (x_1, \emptyset), (x_2, \emptyset), (y_1, \emptyset), (y_2, \emptyset), (t, N_{> 0})\}$.

When playing c from v , the successors are s and t .

W_{Alt} by example – First iteration: playing c from v



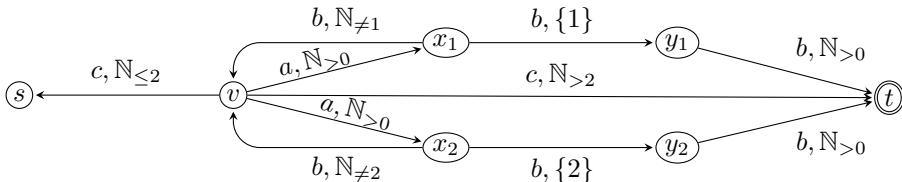
We can win from s when playing
against this number of opponents

Starting with $W_{\text{Alt}}^0 = \{(\overbrace{s, \emptyset}, (v, \emptyset), (x_1, \emptyset), (x_2, \emptyset), (y_1, \emptyset), (y_2, \emptyset), (t, N_{> 0}))\}$.

When playing c from v , the successors are s and t .

- Let us try to reach s : possible when $k \leq 2$. From s , we can win against $\emptyset$ opponents (as $(s, \emptyset) \in W_{\text{Alt}}^0$). So, when going to $\{s\}$ from v , we can win against $N_{\leq 2} \cap \emptyset = \emptyset$ opponents.

W_{Alt} by example – First iteration: playing c from v



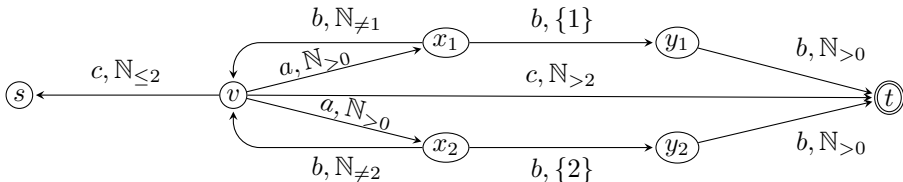
We can win from s when playing against this number of opponents

Starting with $W_{\text{Alt}}^0 = \{(\overbrace{s, \emptyset}, (v, \emptyset), (x_1, \emptyset), (x_2, \emptyset), (y_1, \emptyset), (y_2, \emptyset), (t, N_{>0})\}$.

When playing c from v , the successors are s and t .

- ▶ Let us try to reach s : possible when $k \leq 2$. From s , we can win against $\emptyset$ opponents (as $(s, \emptyset) \in W_{\text{Alt}}^0$). So, when going to $\{s\}$ from v , we can win against $N_{\leq 2} \cap \emptyset = \emptyset$ opponents.
- ▶ Reach t : possible when $k > 2$. From t , we can win against $N_{>0}$ opponents. So, when going to $\{t\}$ from v , we can win against $N_{>2} \cap N_{>0} = N_{>2}$.

W_{Alt} by example – First iteration: playing c from v



We can win from s when playing against this number of opponents

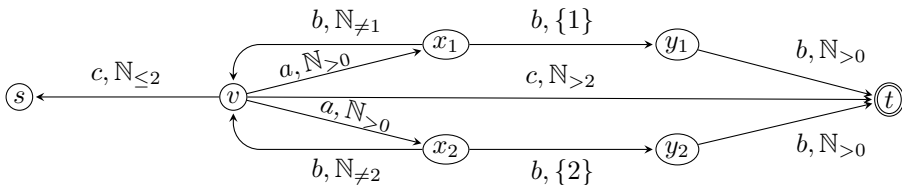
Starting with $W_{\text{Alt}}^0 = \{(\overline{s, \emptyset}), (v, \emptyset), (x_1, \emptyset), (x_2, \emptyset), (y_1, \emptyset), (y_2, \emptyset), (t, N_{>0})\}$.

When playing c from v , the successors are s and t .

- ▶ Let us try to reach s : possible when $k \leq 2$. From s , we can win against $\emptyset$ opponents (as $(s, \emptyset) \in W_{\text{Alt}}^0$). So, when going to $\{s\}$ from v , we can win against $N_{\leq 2} \cap \emptyset = \emptyset$ opponents.
- ▶ Reach t : possible when $k > 2$. From t , we can win against $N_{>0}$ opponents. So, when going to $\{t\}$ from v , we can win against $N_{>2} \cap N_{>0} = N_{>2}$.

So, when playing c from v , we can win if the number of opponents is $\emptyset \cup N_{>2} = N_{>2}$.

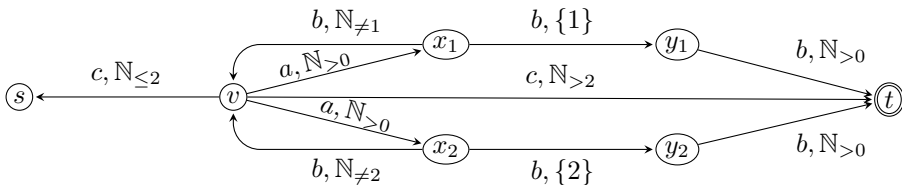
W_{Alt} by example – First iteration



$$W_{\text{Alt}}^0 = \{(s, \emptyset), (v, \emptyset), (x_1, \emptyset), (x_2, \emptyset), (y_1, \emptyset), (y_2, \emptyset), (t, \mathbb{N}_{>0})\}.$$

- ▶ Playing c from v : $\mathbb{N}_{>2}$.
- ▶ Playing a from v : $\emptyset$.
- ▶ Playing b from y_1 : $\mathbb{N}_{>0}$.
- ▶ Playing b from y_2 : $\mathbb{N}_{>0}$.

W_{Alt} by example – First iteration

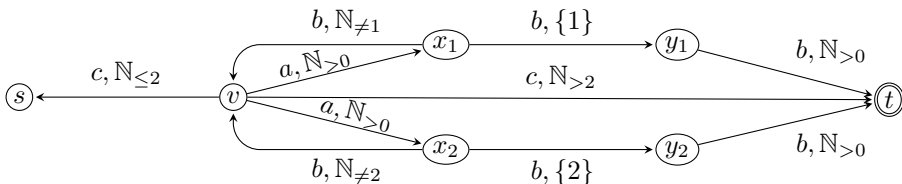


$$W_{\text{Alt}}^0 = \{(s, \emptyset), (v, \emptyset), (x_1, \emptyset), (x_2, \emptyset), (y_1, \emptyset), (y_2, \emptyset), (t, \mathbb{N}_{>0})\}.$$

- ▶ Playing c from v : $\mathbb{N}_{>2}$.
- ▶ Playing a from v : $\emptyset$.
- ▶ Playing b from y_1 : $\mathbb{N}_{>0}$.
- ▶ Playing b from y_2 : $\mathbb{N}_{>0}$.

$$\text{So, } W_{\text{Alt}}^1 = \{(s, \emptyset), (v, \mathbb{N}_{>2}), (x_1, \emptyset), (x_2, \emptyset), (y_1, \mathbb{N}_{>0}), (y_2, \mathbb{N}_{>0}), (t, \mathbb{N}_{>0})\}.$$

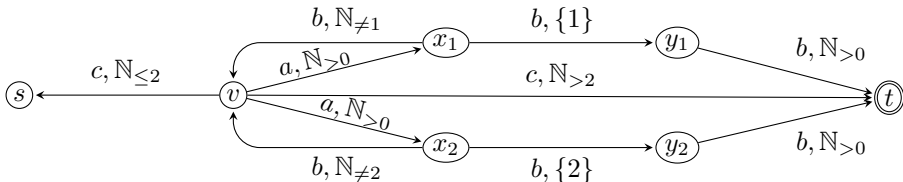
W_{Alt} by example – Second iteration



$$W_{Alt}^1 = \{(s, \emptyset), (v, \mathbb{N}_{>2}), (x_1, \emptyset), (x_2, \emptyset), (y_1, \mathbb{N}_{>0}), (y_2, \mathbb{N}_{>0}), (t, \mathbb{N}_{>0})\}.$$

- ▶ Playing b from x_1 : successors are v and y_1 . So, we can win against $\{1\} \cup \mathbb{N}_{>2} \cup \emptyset = \mathbb{N}_{\neq 2}$ opponents.
- ▶ Playing b from x_2 : win against $\mathbb{N}_{\neq 1}$ opponents.

W_{Alt} by example – Second iteration

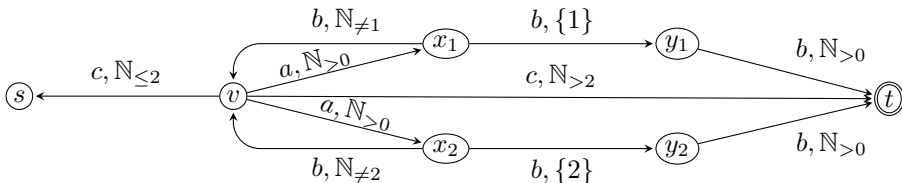


$$W_{\text{Alt}}^1 = \{(s, \emptyset), (v, N_{>2}), (x_1, \emptyset), (x_2, \emptyset), (y_1, N_{>0}), (y_2, N_{>0}), (t, N_{>0})\}.$$

- ▶ Playing b from x_1 : successors are v and y_1 . So, we can win against $\{1\} \cup N_{>2} \cup \emptyset = N_{\neq 2}$ opponents.
- ▶ Playing b from x_2 : win against $N_{\neq 1}$ opponents.

$$\text{So, } W_{\text{Alt}}^2 = \{(s, \emptyset), (v, N_{>2}), (x_1, N_{\neq 2}), (x_2, N_{\neq 1}), (y_1, N_{>0}), (y_2, N_{>0}), (t, N_{>0})\}.$$

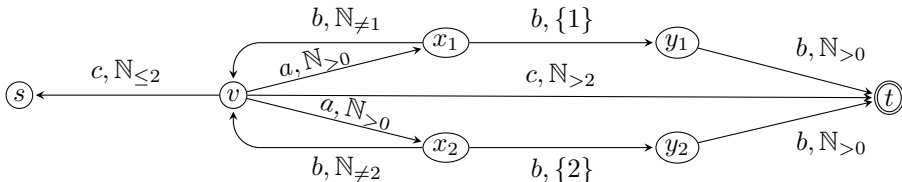
W_{Alt} by example – Third iteration



$$W_{\text{Alt}}^2 = \{(s, \emptyset), (v, N_{>2}), (x_1, N_{\neq 2}), (x_2, N_{\neq 1}), (y_1, N_{>0}), (y_2, N_{>0}), (t, N_{>0})\}.$$

- ▶ Playing a from v : successors are x_1 and x_2 . **But the transition is nondeterministic!** We do not know exactly where we will end up.

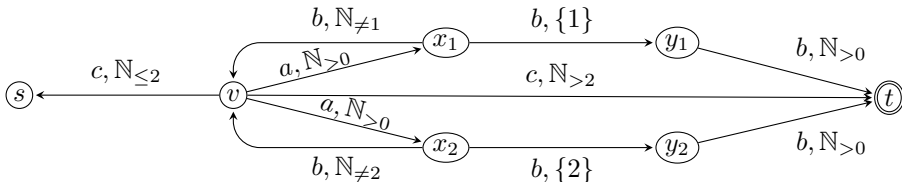
W_{Alt} by example – Third iteration



$$W_{Alt}^2 = \{(s, \emptyset), (v, N_{>2}), (x_1, N_{\neq 2}), (x_2, N_{\neq 1}), (y_1, N_{>0}), (y_2, N_{>0}), (t, N_{>0})\}.$$

- ▶ Playing a from v : successors are x_1 and x_2 . **But the transition is nondeterministic!** We do not know exactly where we will end up.
 - ▶ Go exactly to x_1 : we cannot guarantee it. So, $\emptyset$.
 - ▶ Go exactly to x_2 : we cannot guarantee it. So, $\emptyset$.
 - ▶ Go to either x_1 or x_2 : possible against $N_{>0}$ opponents. Since $(x_1, N_{\neq 2})$ and $(x_2, N_{\neq 1})$, we can win against $N_{\neq 2} \cap N_{\neq 1} = N_{>2}$ opponents.

W_{Alt} by example – Third iteration

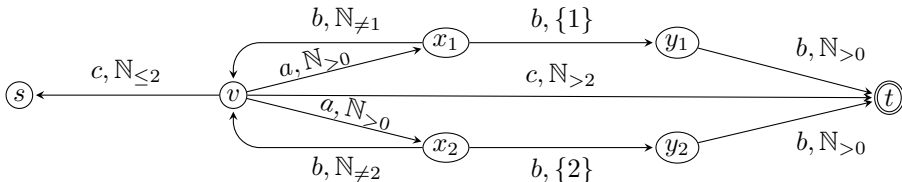


$$W_{\text{Alt}}^2 = \{(s, \emptyset), (v, N_{>2}), (x_1, N_{\neq 2}), (x_2, N_{\neq 1}), (y_1, N_{>0}), (y_2, N_{>0}), (t, N_{>0})\}.$$

- ▶ Playing a from v : successors are x_1 and x_2 . **But the transition is nondeterministic!** We do not know exactly where we will end up.
 - ▶ Go exactly to x_1 : we cannot guarantee it. So, $\emptyset$.
 - ▶ Go exactly to x_2 : we cannot guarantee it. So, $\emptyset$.
 - ▶ Go to either x_1 or x_2 : possible against $N_{>0}$ opponents. Since $(x_1, N_{\neq 2})$ and $(x_2, N_{\neq 1})$, we can win against $N_{\neq 2} \cap N_{\neq 1} = N_{>2}$ opponents.

But $(v, N_{>2}) \in W_{\text{Alt}}^2$. Nothing new under the sun.

W_{Alt} by example – Third iteration



$$W_{\text{Alt}}^2 = \{(s, \emptyset), (v, N_{>2}), (x_1, N_{\neq 2}), (x_2, N_{\neq 1}), (y_1, N_{>0}), (y_2, N_{>0}), (t, N_{>0})\}.$$

- ▶ Playing a from v : successors are x_1 and x_2 . **But the transition is nondeterministic!** We do not know exactly where we will end up.
 - ▶ Go exactly to x_1 : we cannot guarantee it. So, $\emptyset$.
 - ▶ Go exactly to x_2 : we cannot guarantee it. So, $\emptyset$.
 - ▶ Go to either x_1 or x_2 : possible against $N_{>0}$ opponents. Since $(x_1, N_{\neq 2})$ and $(x_2, N_{\neq 1})$, we can win against $N_{\neq 2} \cap N_{\neq 1} = N_{>2}$ opponents.

But $(v, N_{>2}) \in W_{\text{Alt}}^2$. Nothing new under the sun.

$$\text{So, } W_{\text{Alt}}^3 = W_{\text{Alt}}^2.$$

W_{Alt} – Main results

Lemma 8. The sequence $(W_{\text{Alt}}^i)_{i \in \mathbb{N}}$ stabilizes. So, the fixpoint computation terminates.

Corollary 9. Eve has a winning strategy from a vertex v (in the original game)

if and only if

there exists i such that $(v, \mathbb{N}_{>0}) \in W_{\text{Alt}}^i$.

ParaGraphs

Prototype implementation in C++ for the DFS, W, and W_{Alt} algorithms: ParaGraphs³.

Research questions.

- Q1 Between W and W_{Alt} , which approach is the most efficient?
- Q2 Are there arenas on which the antichain-based methods are more efficient than the DFS algorithm?

³<https://gitlab.inria.fr/gastaque/paragraphs>

Results on synthetic arenas

Table 1: Time and memory. Timeout is set to five minutes.

Game		Time (ms)			Memory (kB)		
Name	Value of n	W	W _{Alt}	DFS	W	W _{Alt}	DFS
D-NW-1	11	17312	246	timeout	10365.60	5744.80	timeout
	12	82165	961	timeout	17808.80	7768.00	timeout
	13	timeout	3817	timeout	timeout	11936.80	timeout
D-W-1	11	16696	250	<1	11694.40	6116.40	4686.80
	12	80720	975	<1	19790.80	8058.00	4708.00
	13	timeout	3707	<1	timeout	12733.60	4674.00
D-NW-2	11	15574	85	2	8881.60	4804.80	4716.00
	12	74282	276	3	13774.00	5253.20	4695.60
	13	timeout	945	30	timeout	6324.00	4541.60
D-W-2	11	16680	323	<1	9087.60	5216.80	4725.20
	12	78194	1065	2	14324.00	5982.40	4673.20
	13	timeout	3903	13	timeout	7767.20	4728.40
ND-NW	11	14230	2	timeout	10323.60	4777.60	timeout
	12	67943	6	timeout	16984.40	5126.80	timeout
	13	timeout	9	timeout	timeout	5778.00	timeout

Results on quantified boolean formulas with seven variables

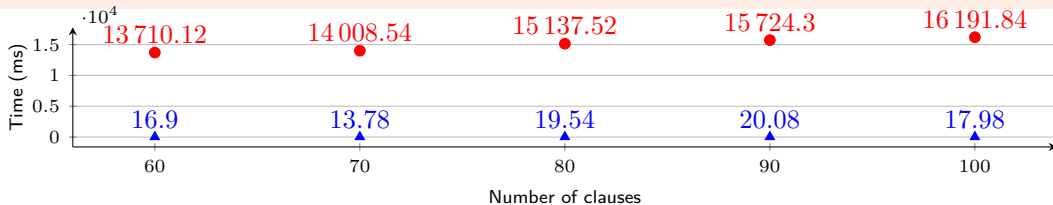


Figure 1: Time taken by W and W_{Alt} on randomly generated QBF instances with seven variables. The DFS algorithm systematically reached the timeout limit. Red circles are for W and blue triangles for W_{Alt} .

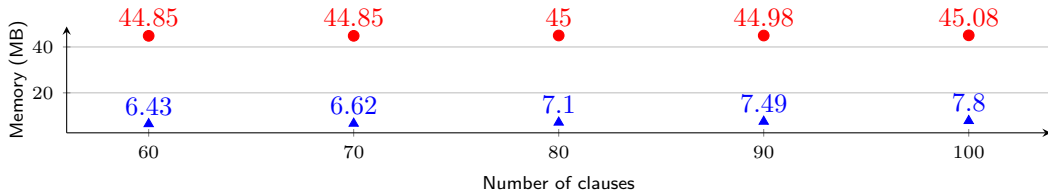


Figure 2: Memory taken by W and W_{Alt} on randomly generated QBF instances with seven variables.

Conclusion

Answers to research questions.

A1 W_{Alt} is more efficient than W , both time- and memory-wise.

A2 There are arenas on which W_{Alt} outspeeds the DFS algorithm.

The memory consumption of the DFS algorithm is always lower than the memory consumption of W_{Alt} .

Our experiments are not sufficient to deduce syntactic characteristics of arenas on which W_{Alt} outperforms the DFS algorithm.

Conclusion

Answers to research questions.

A1 W_{Alt} is more efficient than W , both time- and memory-wise.

A2 There are arenas on which W_{Alt} outspeeds the DFS algorithm.

The memory consumption of the DFS algorithm is always lower than the memory consumption of W_{Alt} .

Our experiments are not sufficient to deduce syntactic characteristics of arenas on which W_{Alt} outperforms the DFS algorithm.

Contributions.

- ▶ Two symbolic algorithms for computing the winning region.
- ▶ Experimental evaluation.
- ▶ (Bonus) a QBF solver.

Conclusion

Answers to research questions.

A1 W_{Alt} is more efficient than W , both time- and memory-wise.

A2 There are arenas on which W_{Alt} outspeeds the DFS algorithm.

The memory consumption of the DFS algorithm is always lower than the memory consumption of W_{Alt} .

Our experiments are not sufficient to deduce syntactic characteristics of arenas on which W_{Alt} outperforms the DFS algorithm.

Contributions.

- ▶ Two symbolic algorithms for computing the winning region.
- ▶ Experimental evaluation.
- ▶ (Bonus) a QBF solver.

Thank you!

Hic sunt dracones

Abandon hope, all ye who enter here. This is the appendix.

Missing definitions on antichains

The **expanded form** of L is obtained by **adding** all the dominated elements:

$$L\downarrow = L \cup \text{Dom}(L).$$

Example 10. $\{\{1, 2\}, \{5\}\}\downarrow = \{\emptyset, \{1\}, \{2\}, \{1, 2\}, \{5\}\}.$

Lemma 11. $L\downarrow = \lceil L \rceil\downarrow$ if and only if $\forall x \in L, \exists y \in \lceil L \rceil : x \sqsubseteq y.$

Antichains for concurrent parameterized games

Recall that

Definition 12. We say that $(v, K) \sqsubseteq (v', K')$ if and only if

- ▶ $v = v'$; and
- ▶ $K \subseteq K'$.

Then, $\sqsubseteq$ is a partial order.

Two main operations:

Join The *join* operation of L and L' is $L \sqcup L' = \lceil L \cup L' \rceil$.

Meet The *meet* operation of L and L' is $L \sqcap L' = \lceil L \downarrow \cap L' \downarrow \rceil$.

W (iterate over $\mathcal{P}(\mathbb{N}_{>0})$)

Definition 13 (W).

$$\text{KPred}[v, a](S) = \{K \in \mathcal{P}(\mathbb{N}_{>0}) \mid \forall v' : (v', K \cap \nabla(v, a, v')) \in S\}$$

$$\text{Pred}(S) = \bigcup_{\text{vertex } v} \bigcup_{\text{action } a} \{(v, K) \mid K \in \text{KPred}[v, a](S)\}.$$

$$W^0 = \{(t, \mathbb{N}_{>0})\} \cup \{(v, \emptyset) \mid v \neq t\}$$

$$\forall i \geq 0 : W^{i+1} = W^i \sqcup \text{Pred}(W^i \downarrow).$$

Lemma 14. For every i , $W^i \downarrow \subseteq W^{i+1} \downarrow$.

Corollary 15. $\lim_{i \rightarrow +\infty} W^i \downarrow$ exists.

Theorem 16. For every i , $W^i = W_{\text{Alt}}^i$.

So, we have **termination for W_{Alt} and W !**

W_{Alt} – Formally

Definition 17 (W_{Alt}).

$$\text{KPred}^{\text{Alt}}[v, a](S) = \left\{ \underbrace{\bigcup_{V' \subseteq V} \left(\underbrace{\nabla(v, a, V')}_{\text{Go exactly from } v \text{ to } V'} \cap \underbrace{\bigcap_{v' \in V'} K_{v'}}_{\text{Can win from } v'} \right)}_{\substack{\text{Can win from } v \\ \text{when going to } V'}} \mid \underbrace{\forall v' : (v', K_{v'}) \in S}_{\text{Handle multiple good numbers of opponents for each } v'} \right\}$$

What are the numbers of opponents against which we can win from v by playing a ?

Can win from v

$$\text{Pred}^{\text{Alt}}(S) = \bigcup_{v \in V} \bigcup_{a \in \Sigma} \{(v, K) \mid K \in \text{KPred}^{\text{Alt}}[v, a](S)\}$$

$$W_{\text{Alt}}^0 = \{(t, \mathbb{N}_{>0})\} \cup \{(v, \emptyset) \mid v \neq t\}$$

$$\forall i \geq 0 : W_{\text{Alt}}^{i+1} = W_{\text{Alt}}^i \sqcup \text{Pred}(W_{\text{Alt}}^i \downarrow)$$

W_{Alt} – More results

Lemma 18. For every i , $W_{\text{Alt}\downarrow}^i \subseteq W_{\text{Alt}\downarrow}^{i+1}$.

Corollary 19. $\lim_{i \rightarrow +\infty} W_{\text{Alt}\downarrow}^i$ exists.

Theorem 20. For every v and K , Eve has a winning strategy from (v, K)

if and only if

there exists i such that $(v, K) \in W_{\text{Alt}\downarrow}^i$.

Lattice $\mathcal{K}$

Definition 21. Let $\mathcal{K}$ be the minimal set such that

- ▶ $\mathbb{N}_{>0} \in \mathcal{K}$, $\emptyset \in \mathcal{K}$,
- ▶ for every $v, v' \in V$ and $a \in \Sigma$, $\nabla(v, a, v') \in \mathcal{K}$, and
- ▶ for every $K, K' \in \mathcal{K}$, one has $K \cup K' \in \mathcal{K}$, $K \cap K' \in \mathcal{K}$, and $K \setminus K' \in \mathcal{K}$,

and let $V_E^{\mathcal{K}} = \{(v, K) \in V_E \mid K \in \mathcal{K}\}$.

Proposition 22. Both $\mathcal{K}$ and $V_E^{\mathcal{K}}$ are finite.

Lemma 23. All values produced by $\text{KPred}^{\text{Alt}}[\cdot](\cdot)$ are in $\mathcal{K}$.
All values produced by $\text{Pred}^{\text{Alt}}(\cdot)$ are in $V_E^{\mathcal{K}}$.

Corollary 24. For every $(v, K) \in W_{\text{Alt}}^i = W^i$, $(v, K) \in V_E^{\mathcal{K}}$.

W in practice

Definition 25 (W). **First, construct $\mathcal{K}$.**

Finite number of sets!

$$\text{KPred}[v, a](S) = \{K \in \mathcal{K} \mid \forall v' : (v', K \cap \nabla(v, a, v')) \in S\}$$

$$\text{Pred}(S) = \bigcup_{\text{vertex } v} \bigcup_{\text{action } a} \{(v, K) \mid K \in \text{KPred}[v, a](S)\}.$$

$$W^0 = \{(t, \mathbb{N}_{>0})\} \cup \{(v, \emptyset) \mid v \neq t\}$$

$$\forall i \geq 0 : W^{i+1} = W^i \sqcup \text{Pred}(W^i \downarrow).$$