Time-optimal Winning Strategies for Poset Games

Martin Zimmermann

RWTH Aachen University

July 17th, 2009

14th International Conference
on Implementation and Application of Automata
Sydney, Australia

Introduction

Two-player games of infinite duration on graphs:

- Solution to the synthesis problem for reactive systems.
- Well-developed theory with nice results.

Introduction

Two-player games of infinite duration on graphs:

- Solution to the synthesis problem for reactive systems.
- Well-developed theory with nice results.

In this talk:

- From linearly ordered objectives to partially ordered objectives.
- Quantitative analysis of winning strategies: synthesize optimal winning strategies.

Introduction

Two-player games of infinite duration on graphs:

- Solution to the synthesis problem for reactive systems.
- Well-developed theory with nice results.

In this talk:

- From linearly ordered objectives to partially ordered objectives.
- Quantitative analysis of winning strategies: synthesize optimal winning strategies.

Outline:

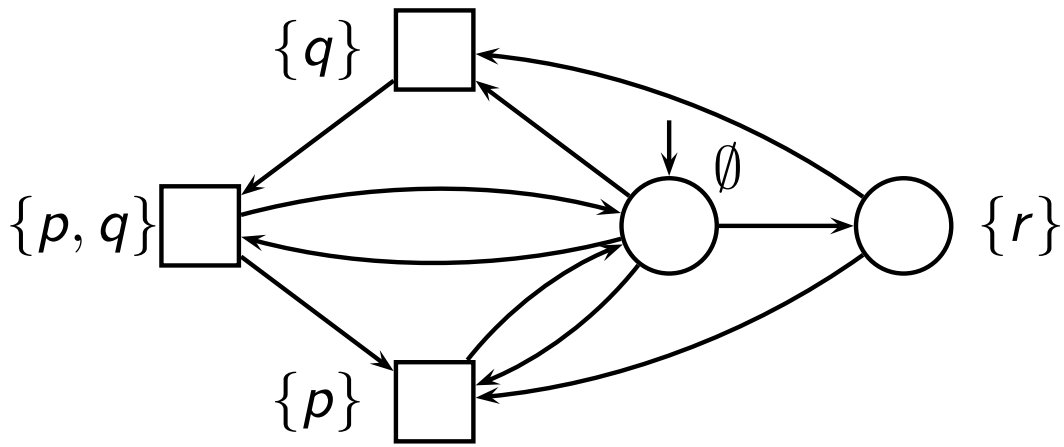
- Definitions and related work
- Poset games
- Time-optimal strategies for poset games

Outline

Definitions

An (initialized and labeled) arena $G = (V, V_0, V_1, E, s_0, l_G)$ consists of

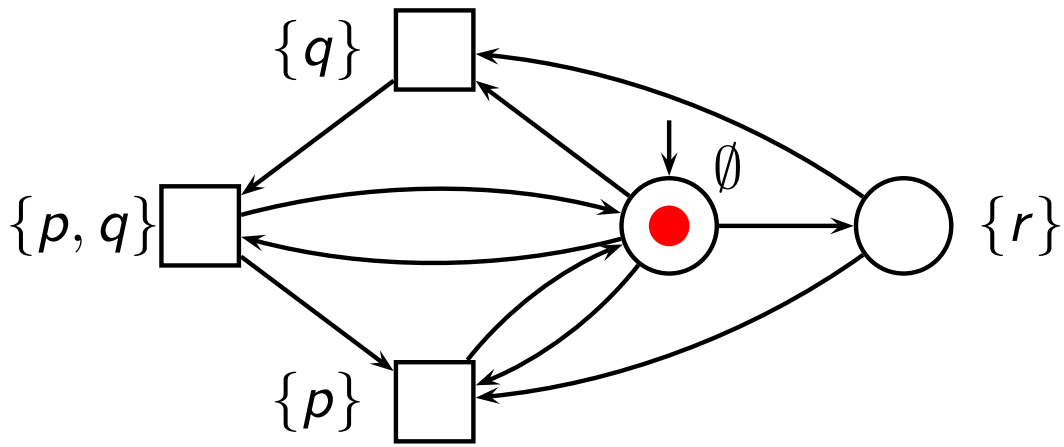
- a finite directed graph (V, E) without dead-ends,
- a partition $\{V_0, V_1\}$ of V denoting the positions of Player 0 (circles) and Player 1 (squares),
- an initial vertex $s_0 \in V$,
- a labeling function $l_G : V \to 2^P$ for some set P of atomic propositions.



Definitions

An (initialized and labeled) arena $G = (V, V_0, V_1, E, s_0, l_G)$ consists of

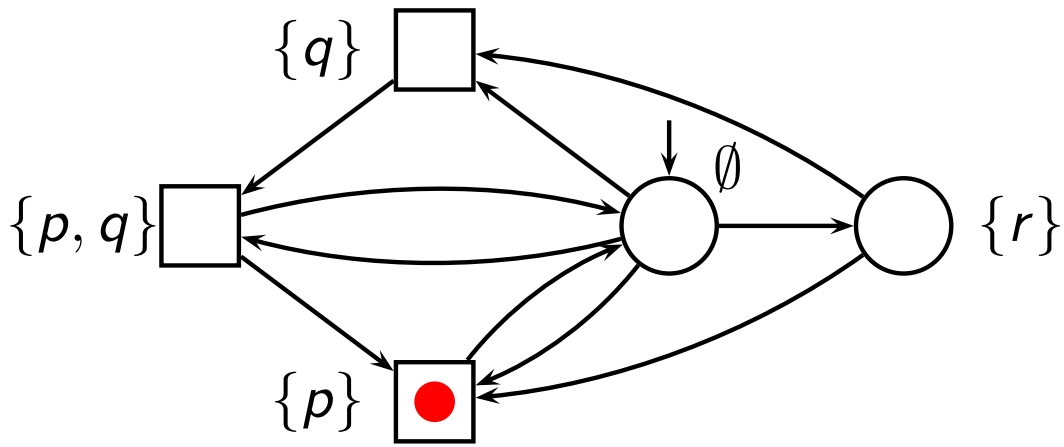
- a finite directed graph (V, E) without dead-ends,
- a partition $\{V_0, V_1\}$ of V denoting the positions of Player 0 (circles) and Player 1 (squares),
- an initial vertex $s_0 \in V$,
- a labeling function $l_G : V \to 2^P$ for some set P of atomic propositions.



Definitions

An (initialized and labeled) arena $G = (V, V_0, V_1, E, s_0, l_G)$ consists of

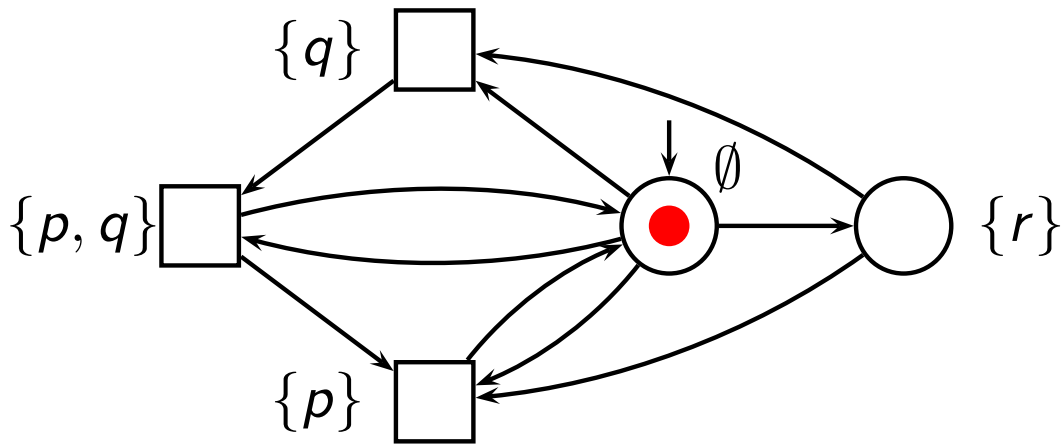
- a finite directed graph (V, E) without dead-ends,
- a partition $\{V_0, V_1\}$ of V denoting the positions of Player 0 (circles) and Player 1 (squares),
- an initial vertex $s_0 \in V$,
- a labeling function $l_G : V \to 2^P$ for some set P of atomic propositions.



Definitions

An (initialized and labeled) arena $G = (V, V_0, V_1, E, s_0, l_G)$ consists of

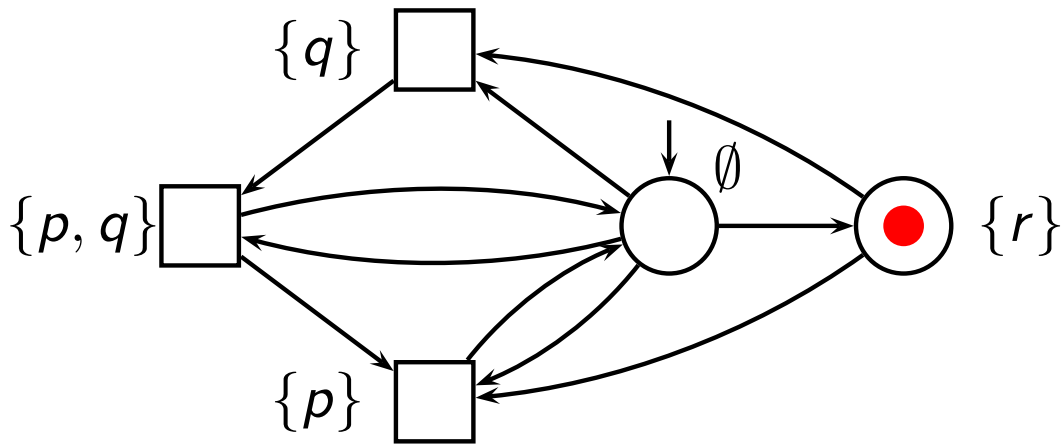
- a finite directed graph (V, E) without dead-ends,
- a partition $\{V_0, V_1\}$ of V denoting the positions of Player 0 (circles) and Player 1 (squares),
- an initial vertex $s_0 \in V$,
- a labeling function $l_G : V \to 2^P$ for some set P of atomic propositions.



Definitions

An (initialized and labeled) arena $G = (V, V_0, V_1, E, s_0, l_G)$ consists of

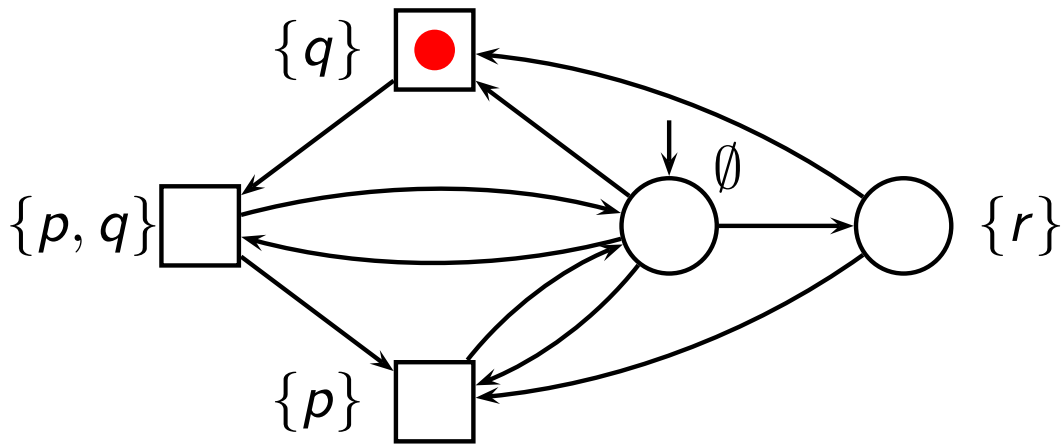
- a finite directed graph (V, E) without dead-ends,
- a partition $\{V_0, V_1\}$ of V denoting the positions of Player 0 (circles) and Player 1 (squares),
- an initial vertex $s_0 \in V$,
- a labeling function $l_G : V \to 2^P$ for some set P of atomic propositions.



Definitions

An (initialized and labeled) arena $G = (V, V_0, V_1, E, s_0, l_G)$ consists of

- a finite directed graph (V, E) without dead-ends,
- a partition $\{V_0, V_1\}$ of V denoting the positions of Player 0 (circles) and Player 1 (squares),
- an initial vertex $s_0 \in V$,
- a labeling function $l_G : V \to 2^P$ for some set P of atomic propositions.



Definitions

- Play in G: infinite path $\rho_0\rho_1\rho_2\ldots$ starting in s_0.
- Strategy for Player i: (partial) mapping $\sigma : V^*V_i \to V$ such that $(s, \sigma(ws)) \in E$; σ is finite-state, if it is computable by a transducer.
- $\rho_0\rho_1\rho_2\ldots$ is consistent with σ: $\rho_{n+1} = \sigma(\rho_0\ldots\rho_n)$ for all n such that $\rho_n \in V_i$.
- Winning plays for Player 0: $\mathrm{Win} \subseteq V^\omega$.
- σ is a winning strategy for Player 0: every play that is consistent with σ is in Win (dual definition for Player 1).

Optimal Strategies

Idea: quantitative analysis of winning strategies.

- The outcome of a play is still binary: win or lose.
- But the (winning) plays are measured according to some evaluation.
- Task: determine optimal (w.r.t. given measure) winning strategies for Player 0.

Optimal Strategies

Idea: quantitative analysis of winning strategies.

- The outcome of a play is still binary: win or lose.
- But the (winning) plays are measured according to some evaluation.
- Task: determine optimal (w.r.t. given measure) winning strategies for Player 0.

Natural measures for some winning conditions:

- Reachability games: time until goal state is visited.
- Büchi games: waiting times between visits of goal states.
- Co-Büchi games: time until goal states are reached for good.

The classical attractor-based algorithms compute optimal strategies for these games.

An Example: Request-Response games

Request-response game: $(G, (Q_j, P_j)_{j=1,\dots,k})$ where $Q_j, P_j \subseteq V$.

- Player 0 wins a play if every visit to Q_j (request) is responded by a later visit to P_j.

An Example: Request-Response games

Request-response game: $(G, (Q_j, P_j)_{j=1,\dots,k})$ where $Q_j, P_j \subseteq V$.

- Player 0 wins a play if every visit to Q_j (request) is responded by a later visit to P_j.
- Waiting times: start a clock for every request that is stopped as soon as it is responded (and ignore subsequent requests).
- Accumulated waiting time: sum up the clock values of a play prefix (quadratic influence of open requests).
- Value of a play: limit superior of the average accumulated waiting time.
- Value of a strategy: value of the worst play consistent with the strategy.

An Example: Request-Response games

Request-response game: $(G, (Q_j, P_j)_{j=1,\dots,k})$ where $Q_j, P_j \subseteq V$.

- Player 0 wins a play if every visit to Q_j (request) is responded by a later visit to P_j.
- Waiting times: start a clock for every request that is stopped as soon as it is responded (and ignore subsequent requests).
- Accumulated waiting time: sum up the clock values of a play prefix (quadratic influence of open requests).
- Value of a play: limit superior of the average accumulated waiting time.
- Value of a strategy: value of the worst play consistent with the strategy.

Theorem (Horn, Thomas, Wallmeier)

If Player 0 has a winning strategy for an RR-game, then she also has an optimal winning strategy, which is finite-state and effectively computable.

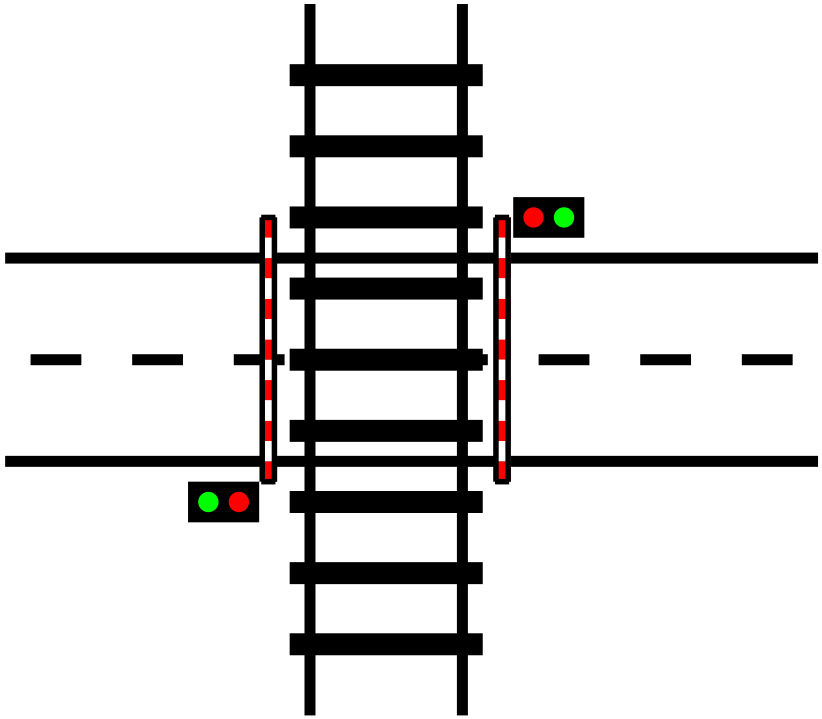
Outline

1. Definitions and Related Work

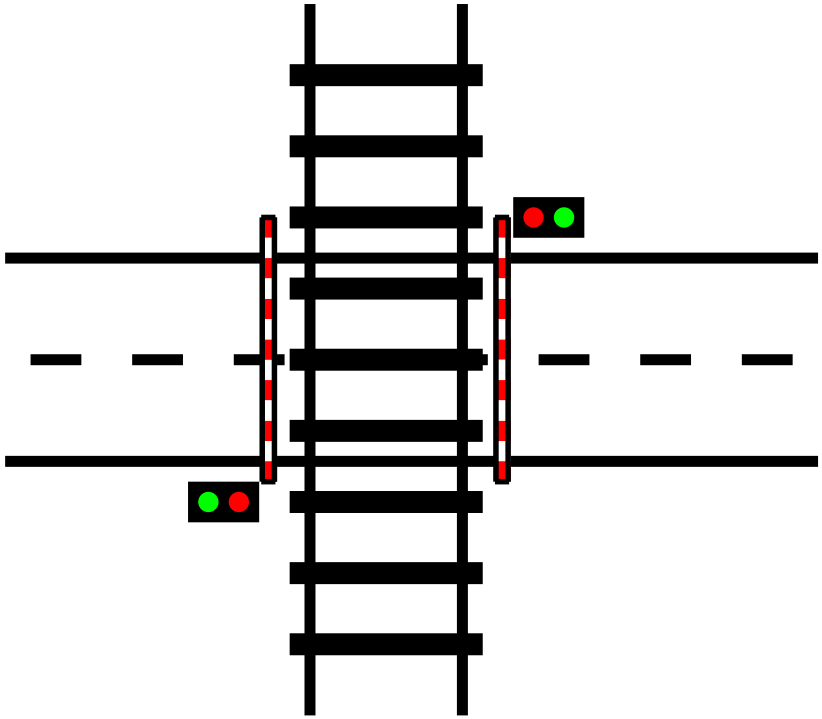
2. **Poset Games**

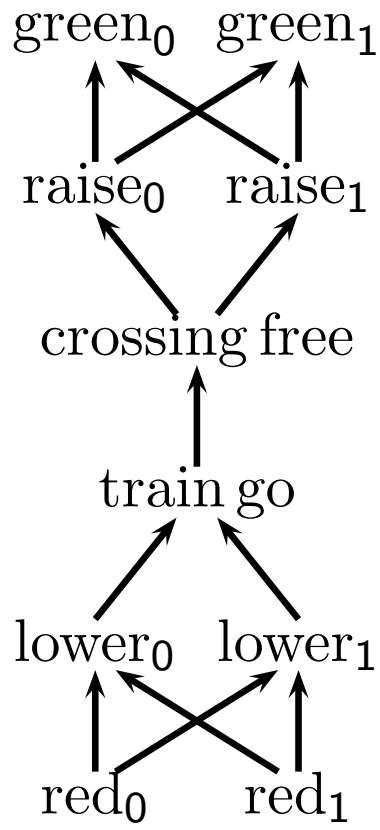
3. Time-optimal Strategies for Poset Games

Motivation for Poset Games

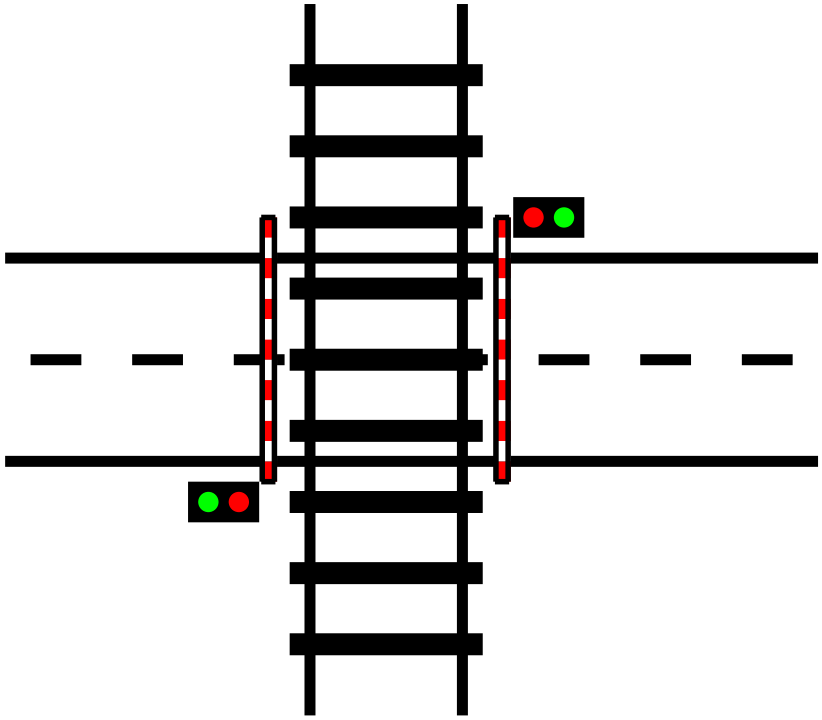


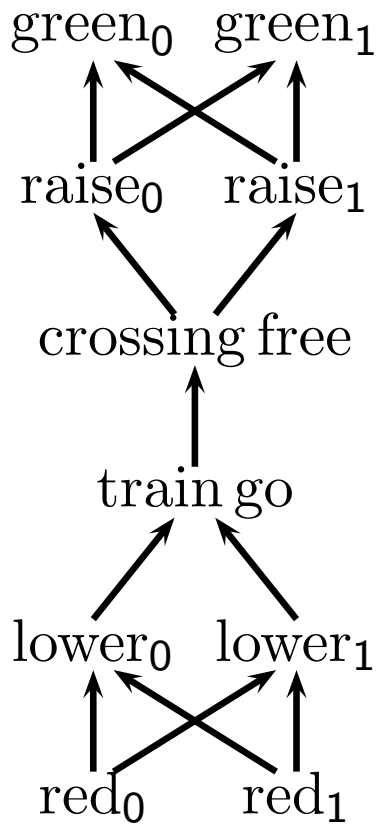
Motivation for Poset Games





Motivation for Poset Games





Generalize RR-games to express more complicated conditions, but retain notion of time-optimality.

Request: still a singular event.

Response: partially ordered set of events.

Poset Games

Poset game: $(G, (q_j, \mathcal{P}_j)_{j=1,\ldots,k})$ where

- G arena (labeled with $l_G : V \to 2^P$),
- $q_j \in P$ request,
- $\mathcal{P}_j = (D_j, \preceq_j)$ poset, where $D_j \subseteq P$.

Remember: P set of atomic propositions

Poset Games

Poset game: $(G, (q_j, \mathcal{P}_j)_{j=1,\ldots,k})$ where

- G arena (labeled with $l_G : V \to 2^P$),
- $q_j \in P$ request,
- $\mathcal{P}_j = (D_j, \preceq_j)$ poset, where $D_j \subseteq P$.

Remember: P set of atomic propositions

Embedding of $\mathcal{P}_j$ in $\rho_0\rho_1\rho_2\ldots$: function $f : D_j \to \mathbb{N}$ such that

- $d \in l_G(\rho_{f(d)})$ for all $d \in D_j$,
- $d \preceq_j d'$ implies $f(d) \leq f(d')$ for all $d, d' \in D_j$.

Poset Games

Poset game: $(G, (q_j, \mathcal{P}_j)_{j=1,\dots,k})$ where

- G arena (labeled with $l_G : V \to 2^P$),
- $q_j \in P$ request,
- $\mathcal{P}_j = (D_j, \preceq_j)$ poset, where $D_j \subseteq P$.

Remember: P set of atomic propositions

Embedding of $\mathcal{P}_j$ in $\rho_0\rho_1\rho_2\dots$: function $f : D_j \to \mathbb{N}$ such that

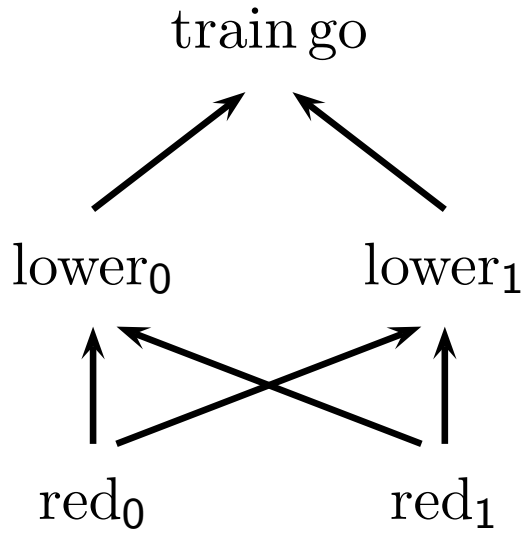
- $d \in l_G(\rho_{f(d)})$ for all $d \in D_j$,
- $d \preceq_j d'$ implies $f(d) \leq f(d')$ for all $d, d' \in D_j$.

Player 0 wins $\rho_0\rho_1\rho_2\dots$ if

$$\forall j \forall n \, (q_j \in l_G(\rho_n) \implies \rho_n\rho_{n+1}\dots \text{ allows embedding of } \mathcal{P}_j) \quad .$$

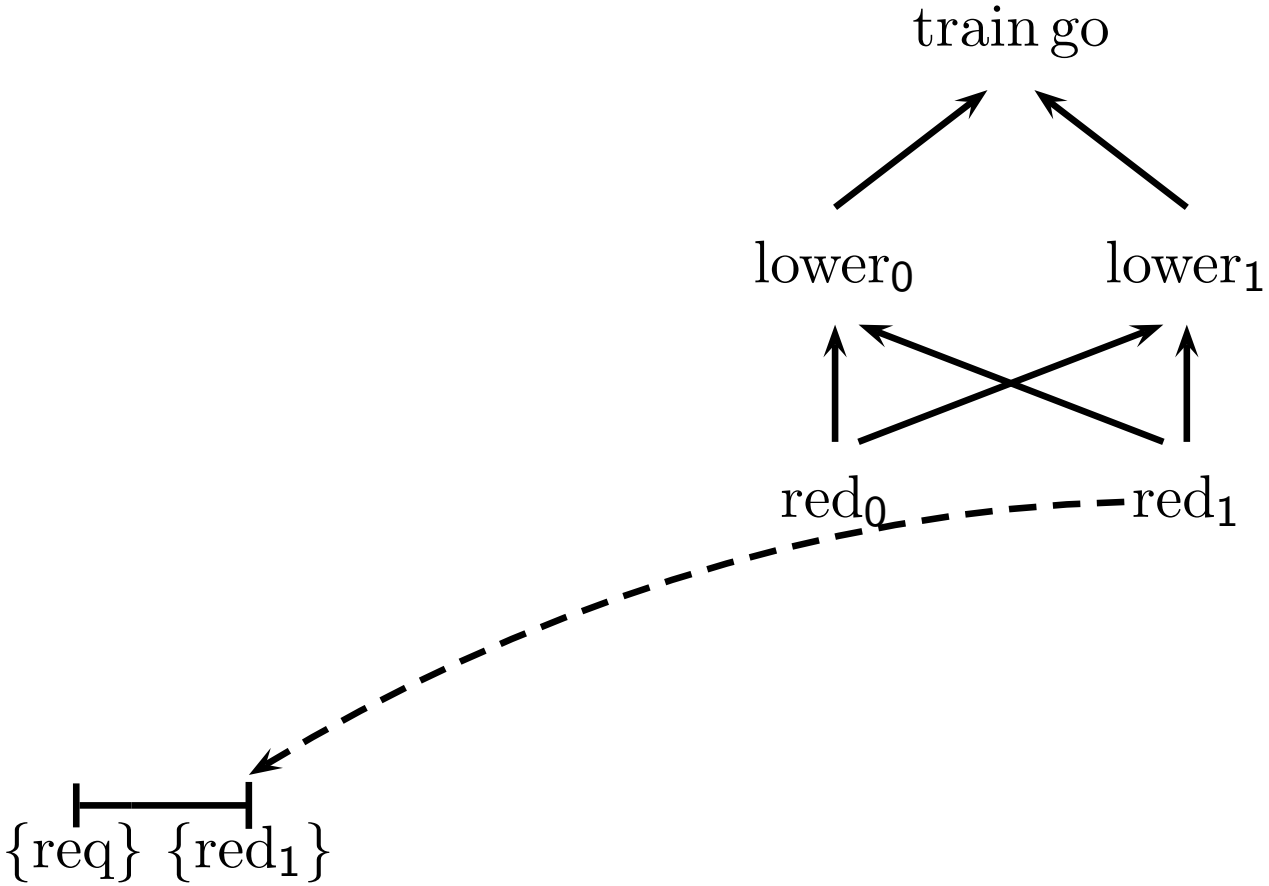
"Every request q_j is responded by a later embedding of $\mathcal{P}_j$."

A Play

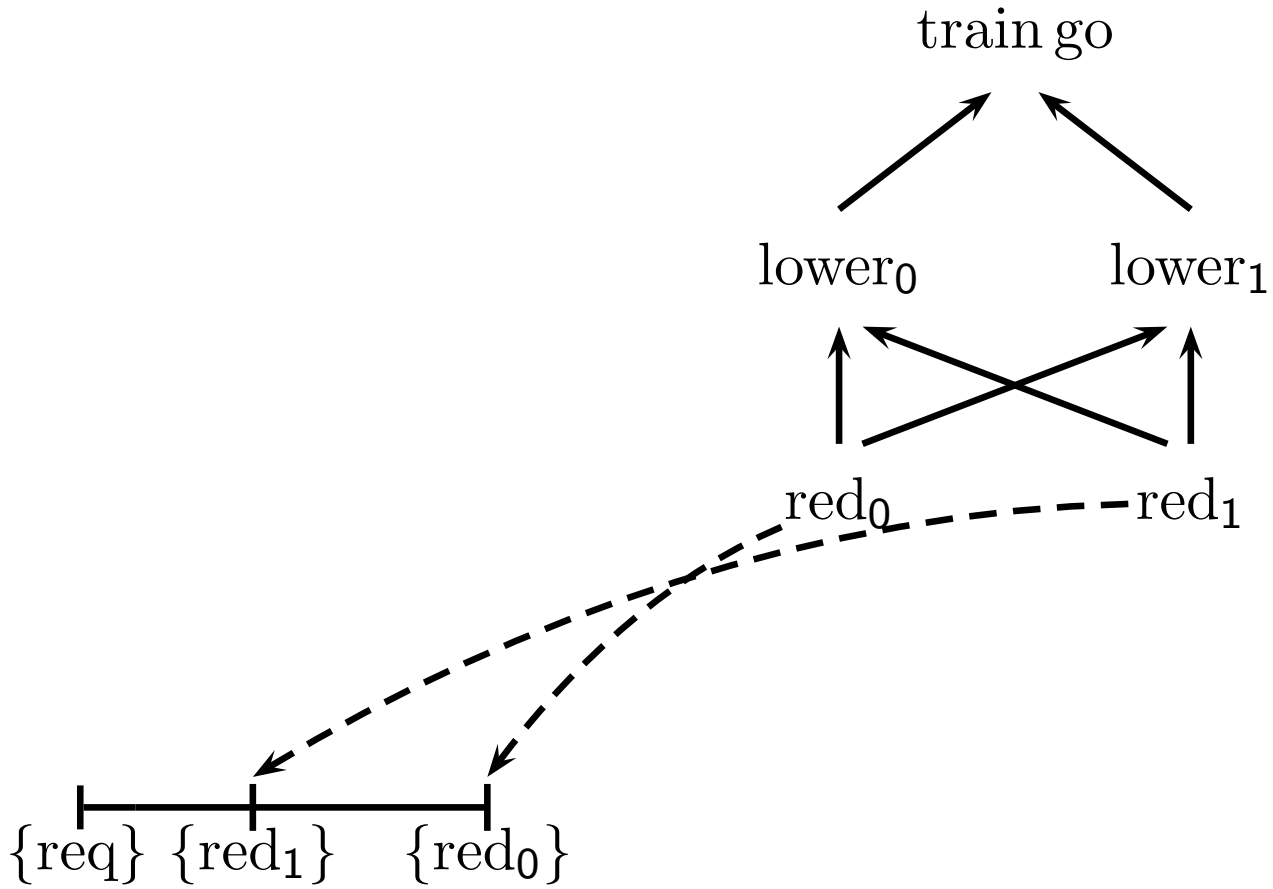


$\{\overline{\text{req}}\}$

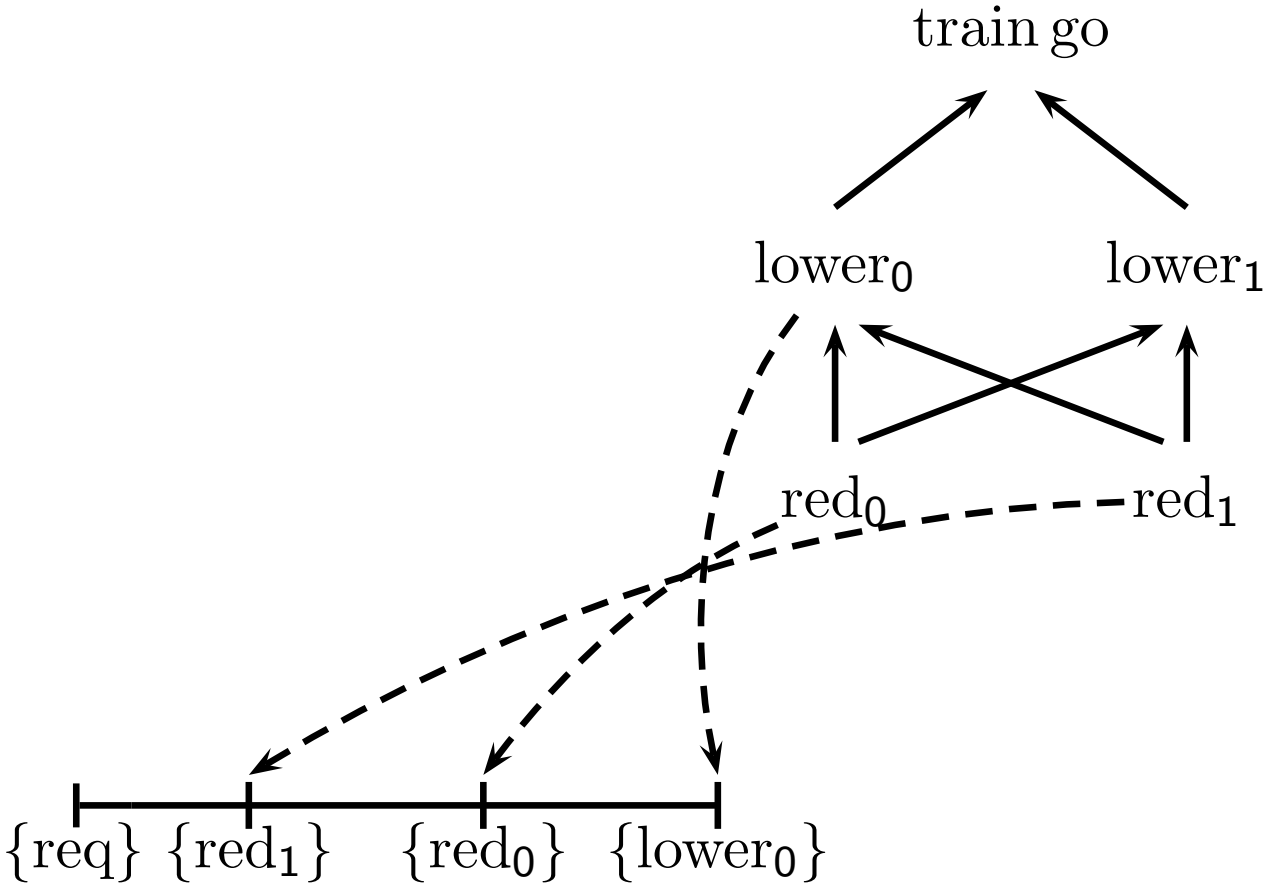
A Play



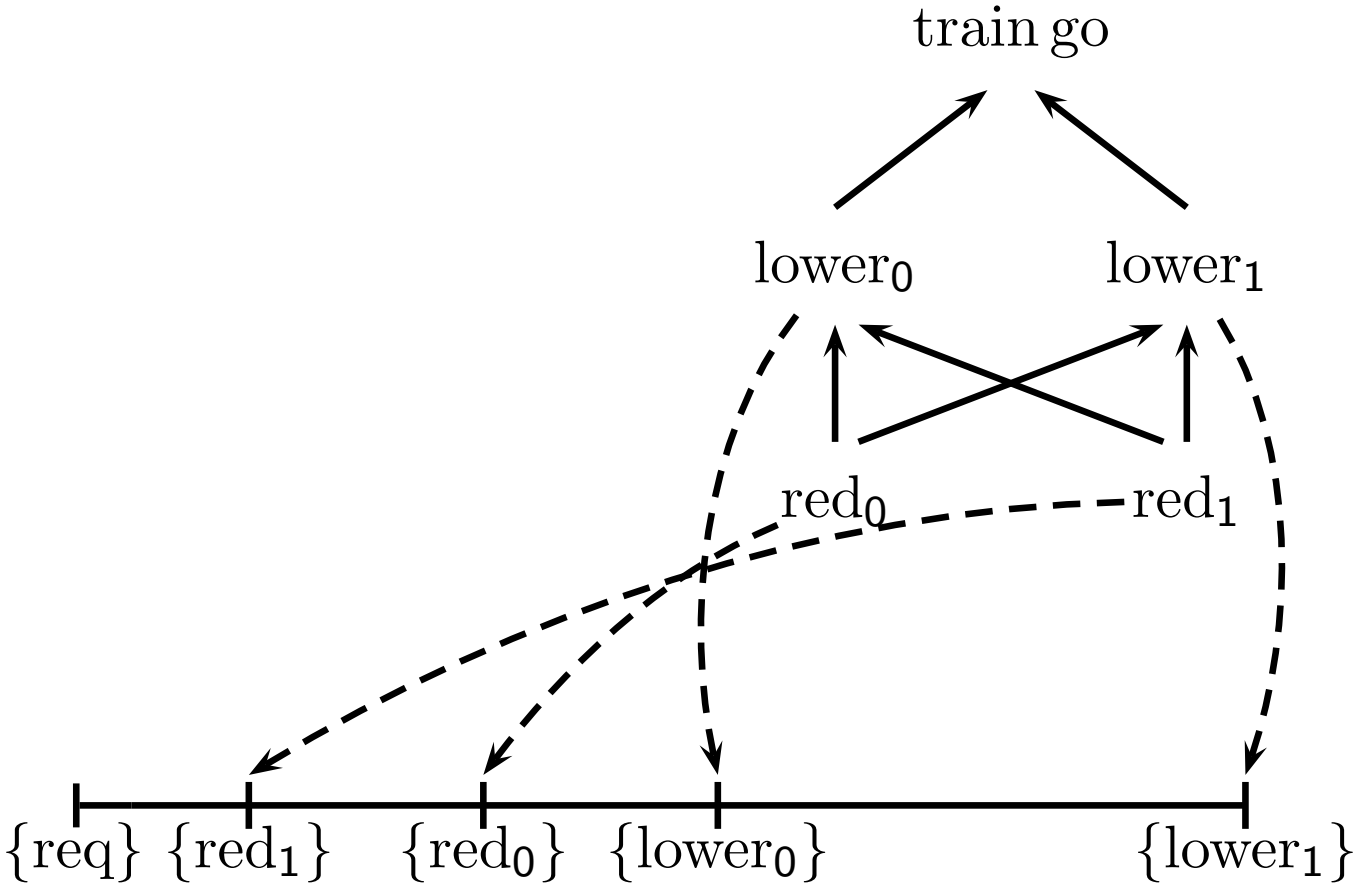
A Play



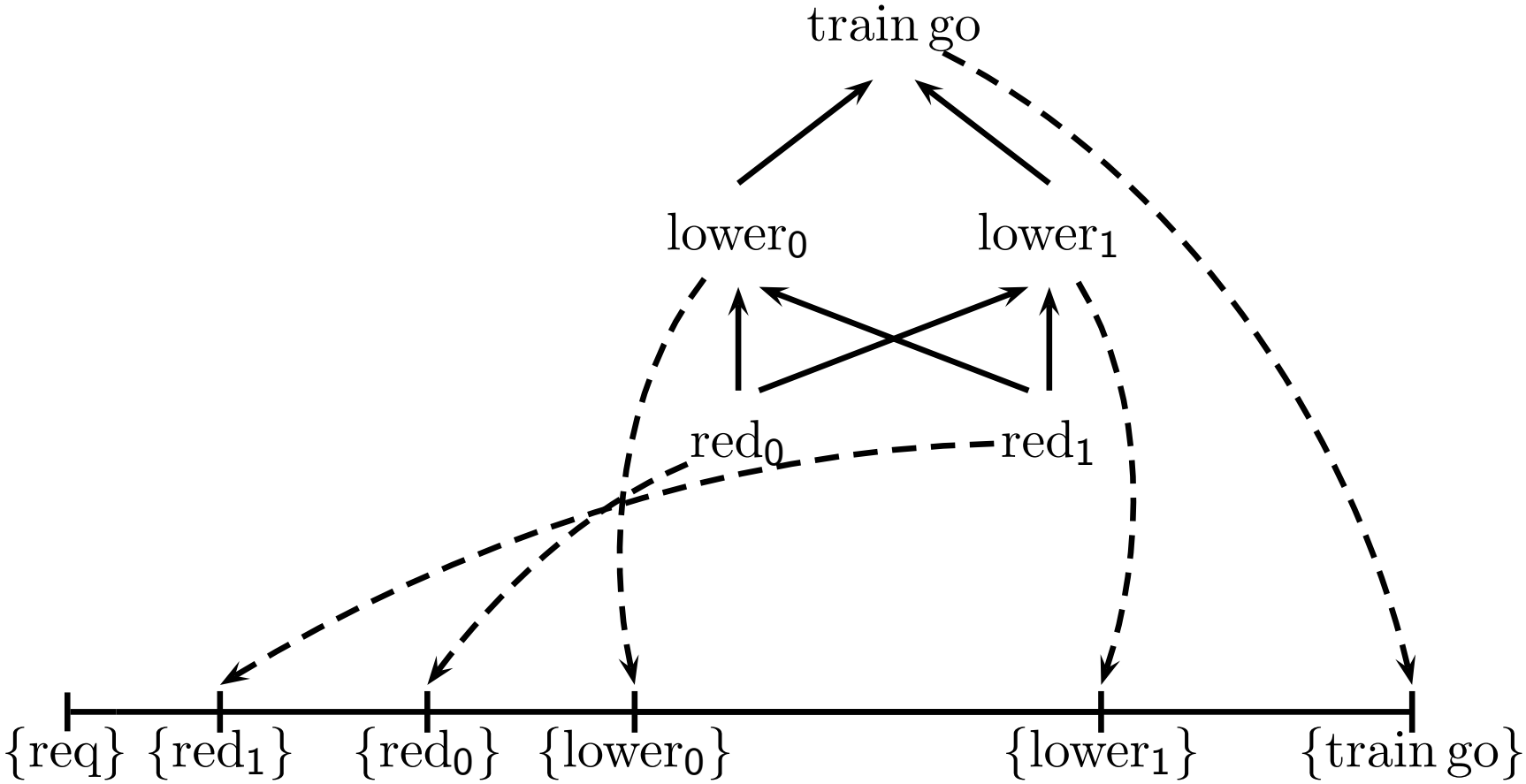
A Play



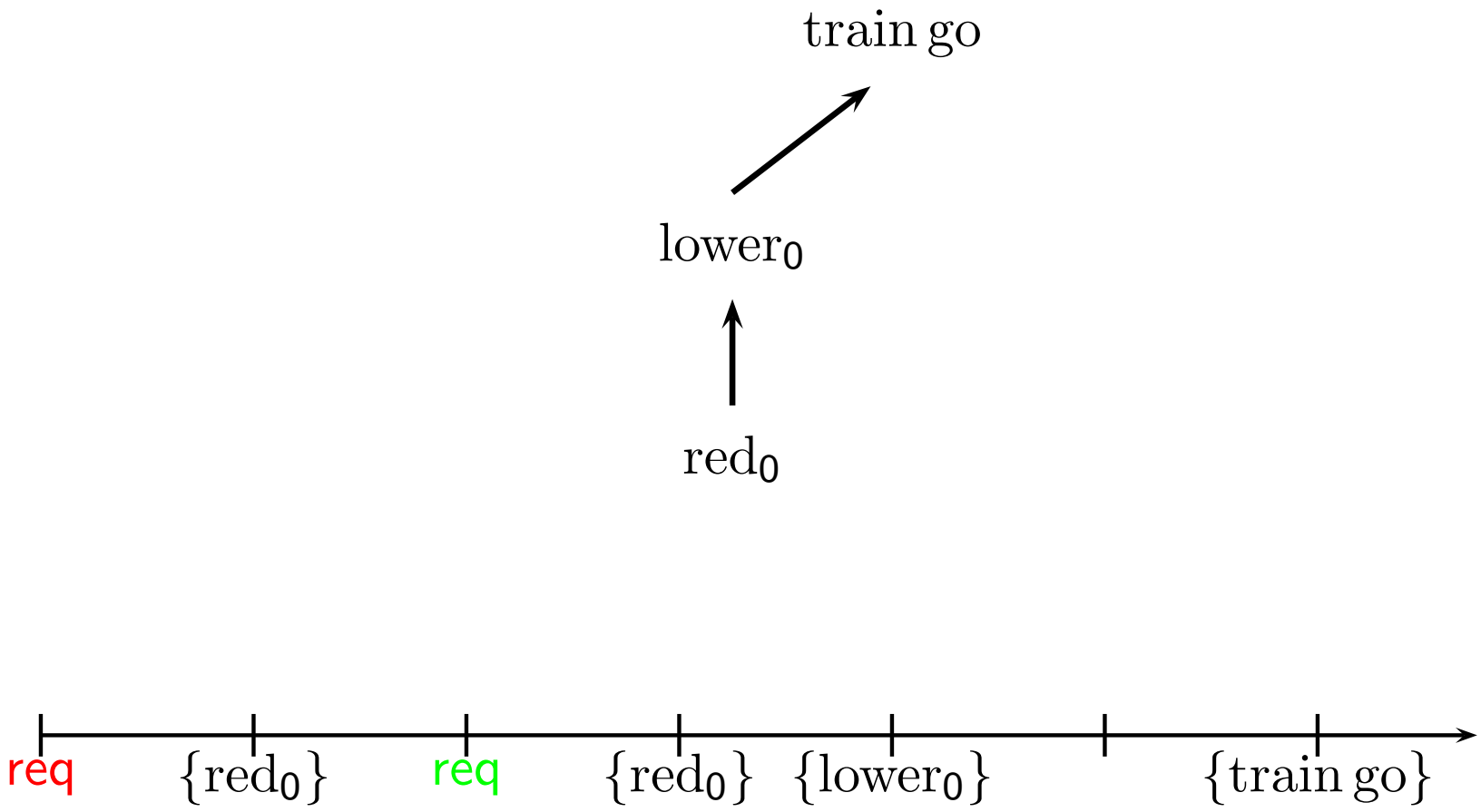
A Play


train go
lower0
lower1
red0
red1
{req}
{red1}
{red0}
{lower0}
{lower1}

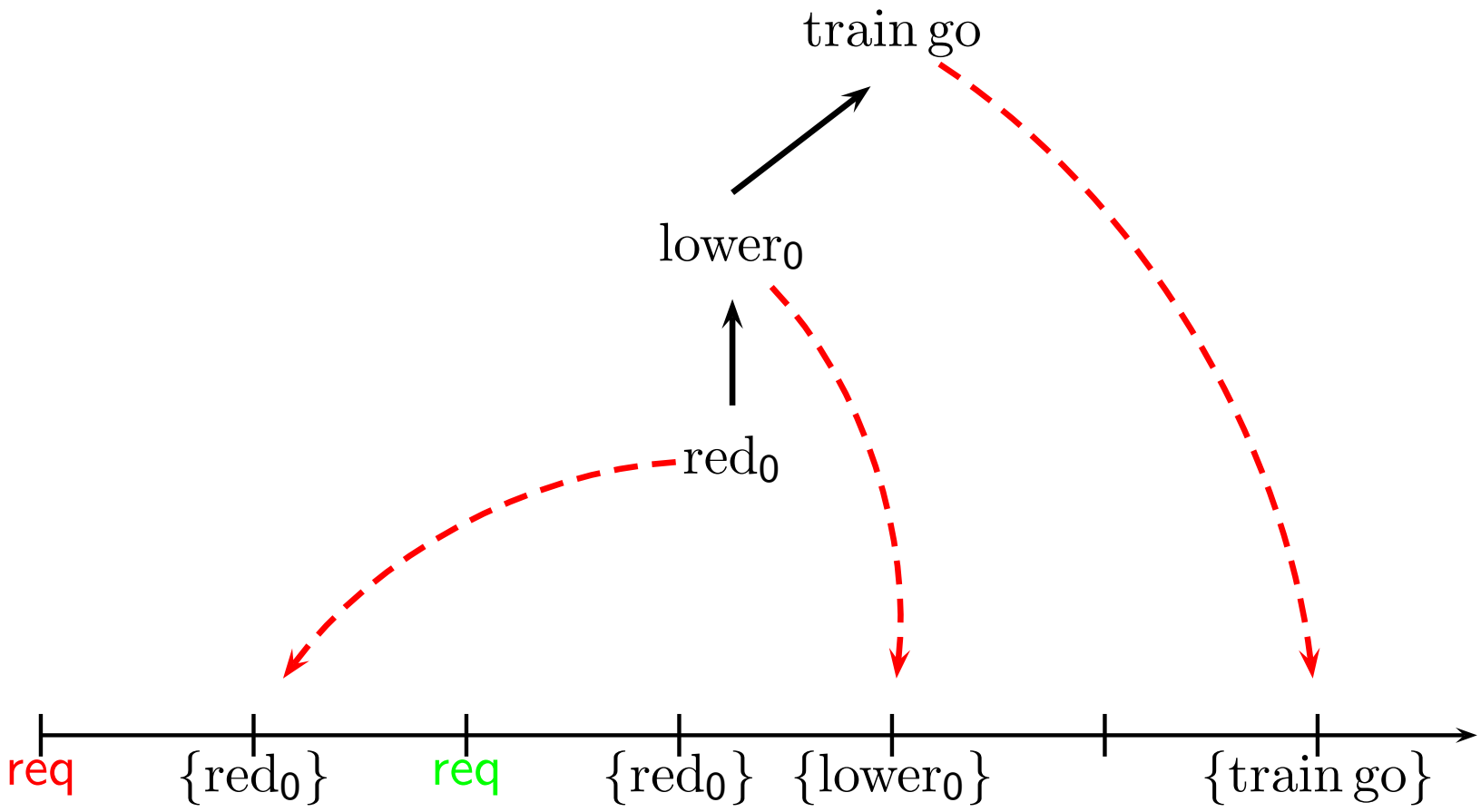
A Play


train go
lower$_0$
lower$_1$
red$_0$
red$_1$
{req}
{red$_1$}
{red$_0$}
{lower$_0$}
{lower$_1$}
{train go}

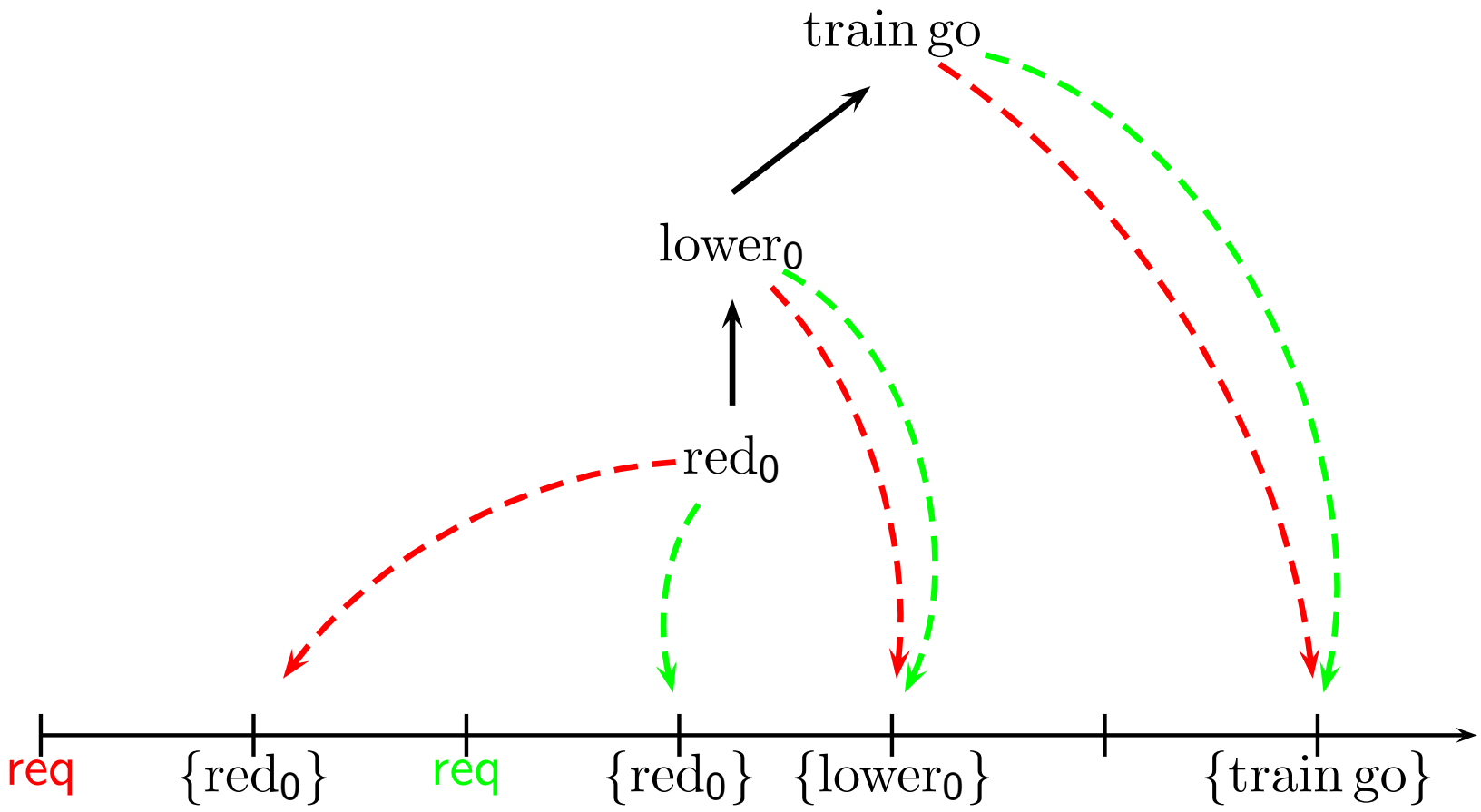
Another Play



Another Play



Another Play



Embeddings might overlap. Hence, we cannot ignore requests that occur while another request is still being responded.

Solving Poset Games

Theorem

Poset games are determined with finite-state strategies, i.e., in every poset games, one of the players has a finite-state winning strategy.

Solving Poset Games

Theorem

Poset games are determined with finite-state strategies, i.e., in every poset games, one of the players has a finite-state winning strategy.

Proof:
Reduction to Büchi games; memory is used

- to store elements of the posets that still have to be embedded,
- to deal with overlapping embeddings,
- to implement a cyclic counter to ensure that every request is responded by an embedding.

Size of the memory: exponential in the size of the posets $\mathcal{P}_j$.

Outline

1. Definitions and Related Work

2. Poset Games

3. **Time-optimal Strategies for Poset Games**

Waiting Times

As desired, a natural definition of waiting times is retained:

- Start a clock if a request is encountered...
- ... that is stopped as soon as the embedding is completed.
- Need a clock for every request (even if another request is already open).

Waiting Times

As desired, a natural definition of waiting times is retained:

- Start a clock if a request is encountered...
- ... that is stopped as soon as the embedding is completed.
- Need a clock for every request (even if another request is already open).
- Value of a play: limit superior of the average accumulated waiting time.
- Value of a strategy: value of the worst play consistent with the strategy.
- Corresponding notion of optimal strategies.

The Main Theorem

Theorem

If Player 0 *has a winning strategy for a poset game* $\mathcal{G}$*, then she also has an optimal winning strategy, which is finite-state and effectively computable.*

The Main Theorem

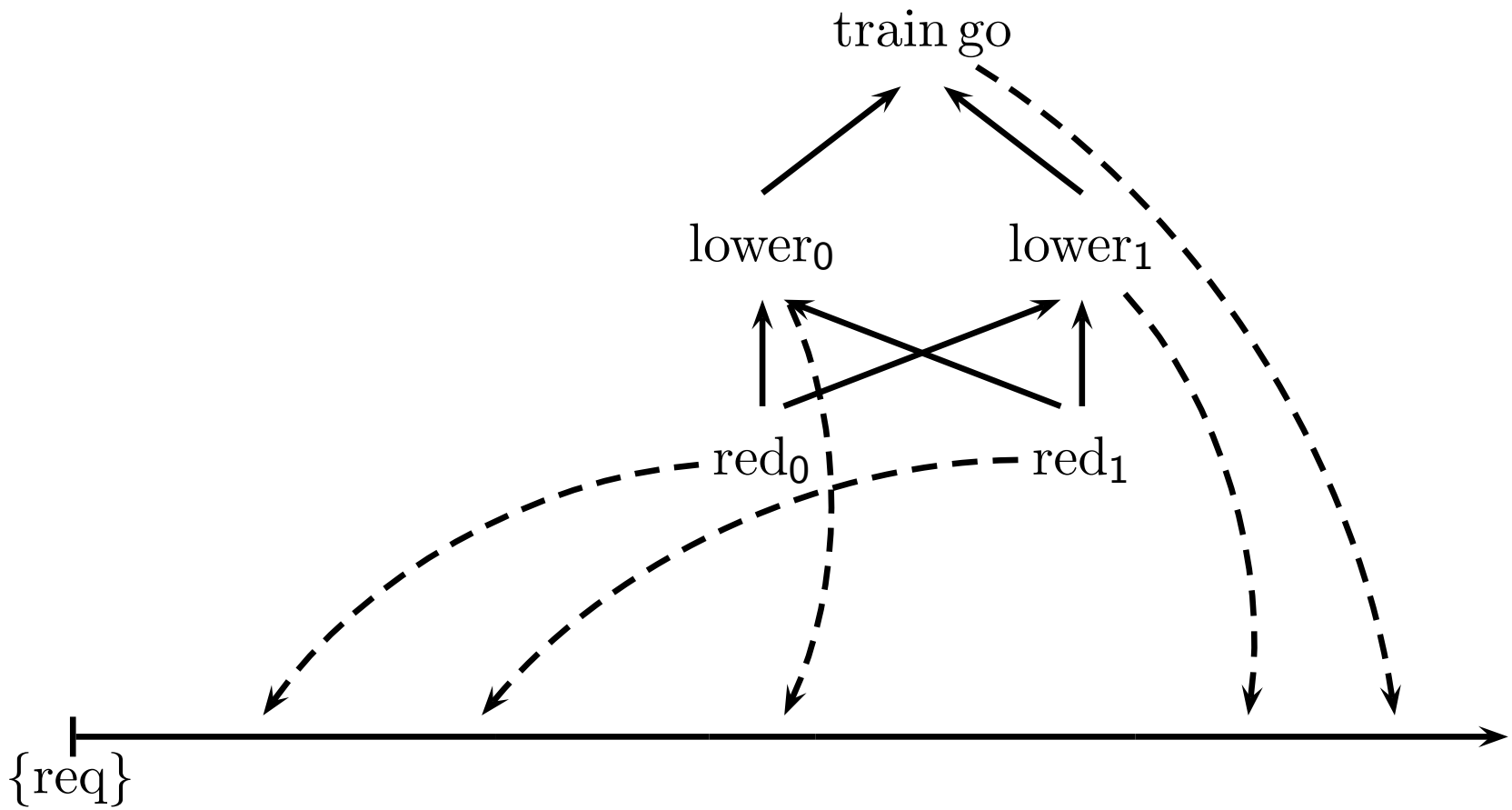
Theorem

If Player 0 has a winning strategy for a poset game $\mathcal{G}$, then she also has an optimal winning strategy, which is finite-state and effectively computable.

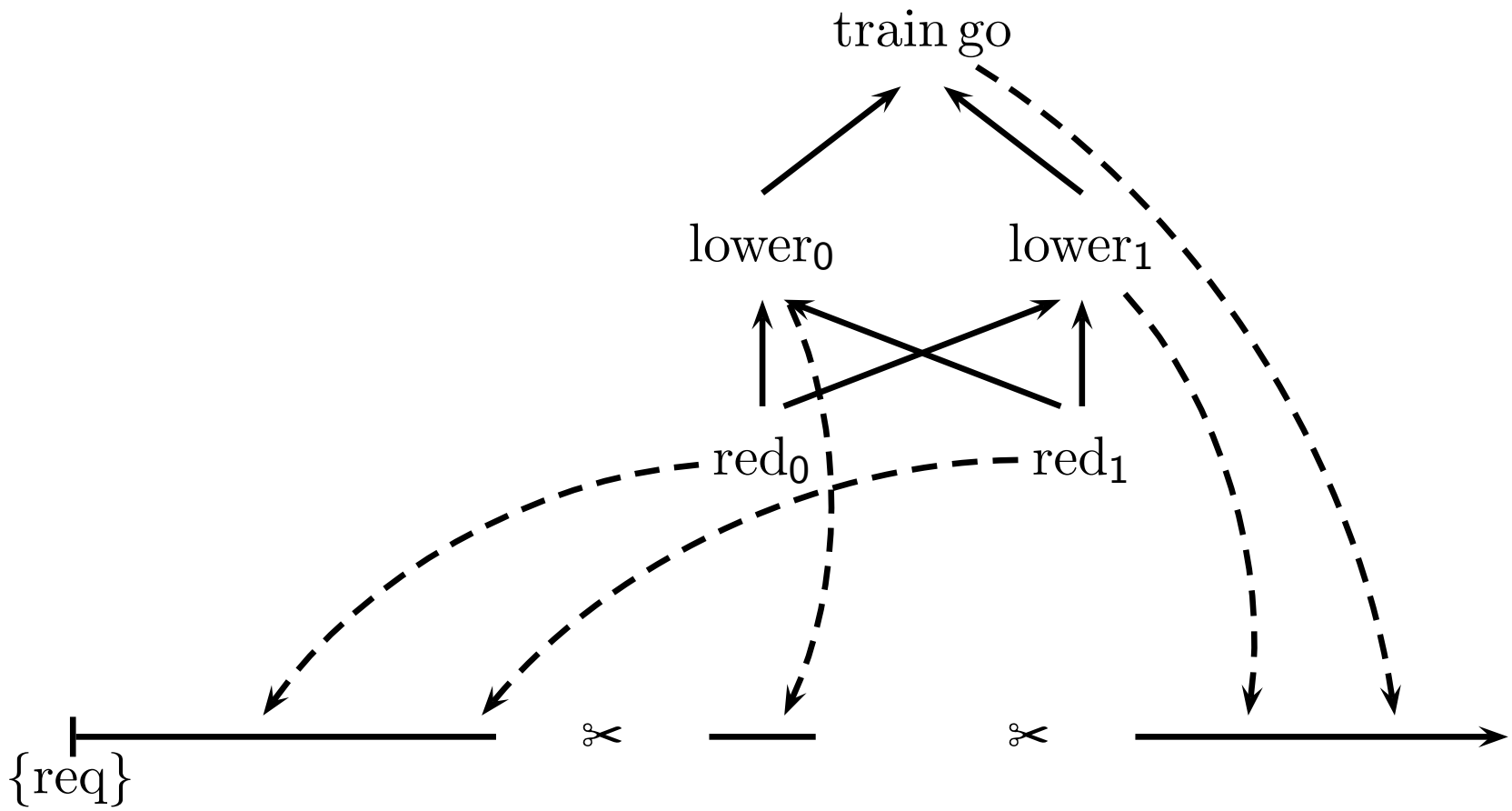
Proof:

- If Player 0 has a winning strategy, then she also has one of value less than a certain constant c (from reduction). This bounds the value of an optimal strategy, too.
- For every strategy of value $\leq c$ there is another strategy of smaller or equal value, that also bounds all waiting times and bounds the number of open requests.
- If the waiting times and the number of open requests are bounded, then $\mathcal{G}$ can be reduced to a mean-payoff game.

Proof: Bounding the Waiting Times

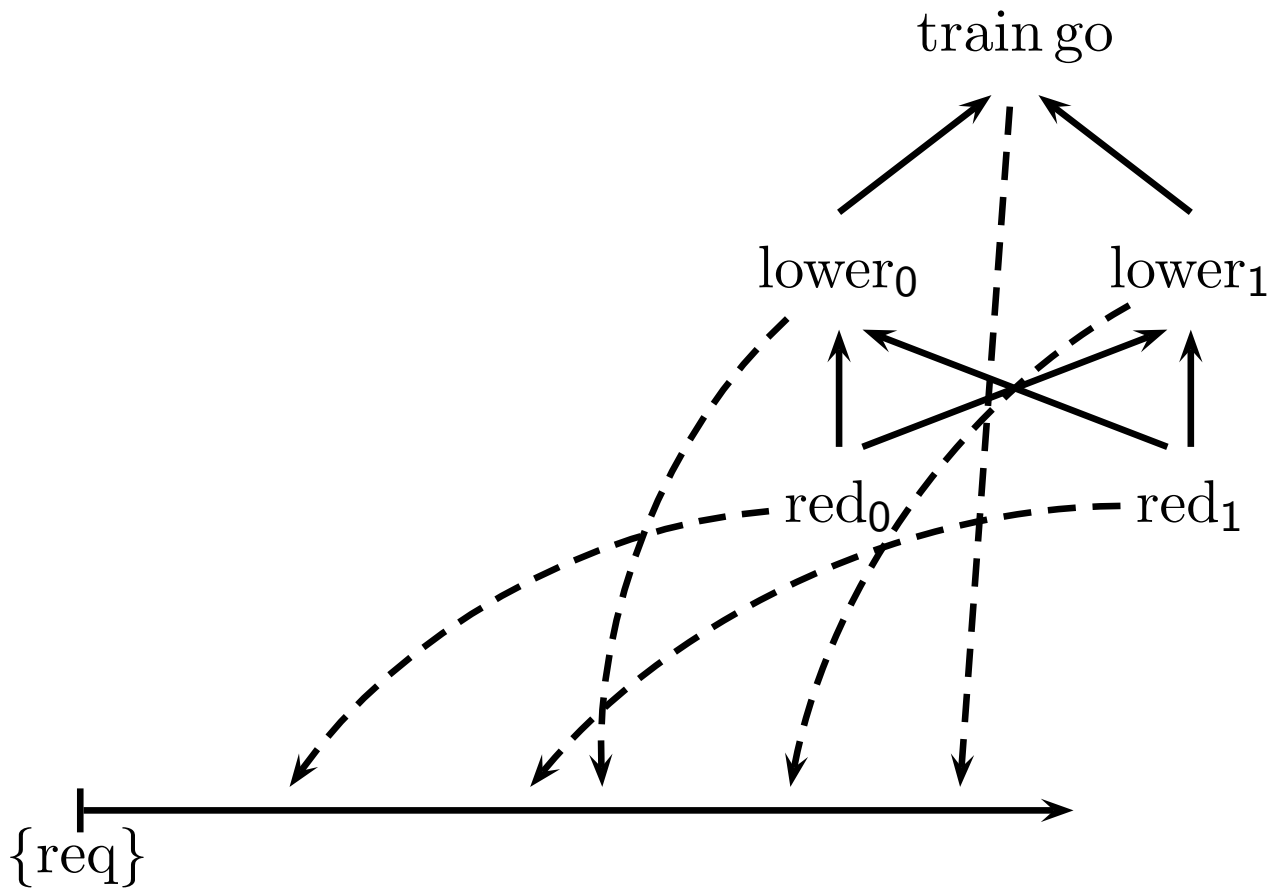


Proof: Bounding the Waiting Times



Skip loops, but pay attention to other overlapping embeddings!

Proof: Bounding the Waiting Times



Repeating this leads to bounded waiting times and a bounded number of open requests.

Proof: Reduction to Mean-Payoff Games

Mean-payoff game: (G, l) where the edges of the arena G are labeled by $l : E \to \mathbb{Z}$.

- goal for Player 0: maximize the limit inferior of the average accumulated edge labels.
- goal for Player 1: minimize the limit superior of the average accumulated edge labels.

Proof: Reduction to Mean-Payoff Games

Mean-payoff game: (G, l) where the edges of the arena G are labeled by $l : E \to \mathbb{Z}$.

- goal for Player 0: maximize the limit inferior of the average accumulated edge labels.
- goal for Player 1: minimize the limit superior of the average accumulated edge labels.

Theorem (Ehrenfeucht, Mycielski / Zwick, Paterson)

In a mean-payoff game, both players have optimal strategies, which are positional and effectively computable.

Proof: Reduction to Mean Payoff Games

From a poset game $\mathcal{G}$ with bounded waiting times and a bounded number of open requests, construct a mean-payoff game $\mathcal{G}'$ such that

- the memory keeps track of the waiting times, and
- the value of a play in $\mathcal{G}$ and the payoff for Player 1 for the corresponding play in $\mathcal{G}'$ are equal.

Then: an optimal strategy for Player 1 in $\mathcal{G}'$ induces an optimal strategy for Player 0 in $\mathcal{G}$.

Proof: Reduction to Mean Payoff Games

From a poset game $\mathcal{G}$ with bounded waiting times and a bounded number of open requests, construct a mean-payoff game $\mathcal{G}'$ such that

- the memory keeps track of the waiting times, and
- the value of a play in $\mathcal{G}$ and the payoff for Player 1 for the corresponding play in $\mathcal{G}'$ are equal.

Then: an optimal strategy for Player 1 in $\mathcal{G}'$ induces an optimal strategy for Player 0 in $\mathcal{G}$.

Size of the mean-payoff game: super-exponential in the size of the poset game (holds already for RR-games).

Conclusion

We introduced a new winning condition for infinite games

- that extends the request-response condition,
- is well-suited to model real-life requirements,
- but retains a natural definition of waiting times and optimal strategies.

We showed the existence of optimal strategies for poset games, which are finite-state and effectively computable.

Conclusion

We introduced a new winning condition for infinite games

- that extends the request-response condition,
- is well-suited to model real-life requirements,
- but retains a natural definition of waiting times and optimal strategies.

We showed the existence of optimal strategies for poset games, which are finite-state and effectively computable.

Further research:

- Avoid the detour via mean-payoff games and directly compute (approximatively) optimal strategies, to overcome the atrocious complexity.
- Understand the trade-off between the size and value of a strategy.
- Consider optimal strategies for other winning conditions.