Parametric Linear Temporal Logics

Joint work with Peter Faymonville, Florian Horn,
Wolfgang Thomas, and Nico Wallmeier

Martin Zimmermann

Saarland University

March 10th, 2015

Aalborg University, Aalborg, Denmark

Motivation

Linear Temporal Logic (LTL) as specification language:

- Simple and variable-free syntax and intuitive semantics.
- Expressively equivalent to first-order logic on words.
- LTL model checking routinely applied in industrial settings.

Motivation

Linear Temporal Logic (LTL) as specification language:

- Simple and variable-free syntax and intuitive semantics.
- Expressively equivalent to first-order logic on words.
- LTL model checking routinely applied in industrial settings.

Shortcomings:

1. LTL cannot express timing constraints.

Motivation

Linear Temporal Logic (LTL) as specification language:

- Simple and variable-free syntax and intuitive semantics.
- Expressively equivalent to first-order logic on words.
- LTL model checking routinely applied in industrial settings.

Shortcomings:

1. LTL cannot express timing constraints.

2. LTL cannot express all ω-regular properties.

Motivation

Linear Temporal Logic (LTL) as specification language:

- Simple and variable-free syntax and intuitive semantics.
- Expressively equivalent to first-order logic on words.
- LTL model checking routinely applied in industrial settings.

Shortcomings:

1. LTL cannot express timing constraints.
   - Add $\mathbf{F}_{\leq k}$ for $k \in \mathbb{N}$.
2. LTL cannot express all ω-regular properties.

Motivation

Linear Temporal Logic (LTL) as specification language:

- Simple and variable-free syntax and intuitive semantics.
- Expressively equivalent to first-order logic on words.
- LTL model checking routinely applied in industrial settings.

Shortcomings:

1. LTL cannot express timing constraints.
   - Add $\mathbf{F}_{\leq k}$ for $k \in \mathbb{N}$. Not practical: how to determine appropriate k.
2. LTL cannot express all ω-regular properties.

Motivation

Linear Temporal Logic (LTL) as specification language:

- Simple and variable-free syntax and intuitive semantics.
- Expressively equivalent to first-order logic on words.
- LTL model checking routinely applied in industrial settings.

Shortcomings:

1. LTL cannot express timing constraints.
   - Add $\mathbf{F}_{\leq k}$ for $k \in \mathbb{N}$. Not practical: how to determine appropriate k.
   - Add $\mathbf{F}_{\leq x}$ for variable x.

2. LTL cannot express all ω-regular properties.

Motivation

Linear Temporal Logic (LTL) as specification language:

- Simple and variable-free syntax and intuitive semantics.
- Expressively equivalent to first-order logic on words.
- LTL model checking routinely applied in industrial settings.

Shortcomings:

1. LTL cannot express timing constraints.
   - Add $\mathbf{F}_{\leq k}$ for $k \in \mathbb{N}$. Not practical: how to determine appropriate k.
   - Add $\mathbf{F}_{\leq x}$ for variable x. Now: does there exist a valuation for x s.t. specification is satisfied?
2. LTL cannot express all ω-regular properties.

Motivation

Linear Temporal Logic (LTL) as specification language:

- Simple and variable-free syntax and intuitive semantics.
- Expressively equivalent to first-order logic on words.
- LTL model checking routinely applied in industrial settings.

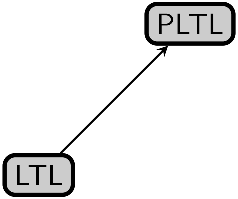
Shortcomings:

1. LTL cannot express timing constraints.
   - Add $\mathbf{F}_{\leq k}$ for $k \in \mathbb{N}$. Not practical: how to determine appropriate k.
   - Add $\mathbf{F}_{\leq x}$ for variable x. Now: does there exist a valuation for x s.t. specification is satisfied?
2. LTL cannot express all ω-regular properties.
   - Many extensions that are equivalent to ω-regular languages: add regular expression-, grammar-, or automata-operators to LTL.

Overview

LTL

Overview



Parametric LTL

Alur et al. '99: add parameterized operators to LTL

$$\varphi ::= p \mid \neg p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \mathbf{X}\varphi \mid \varphi \mathbf{U} \varphi \mid \varphi \mathbf{R} \varphi \mid \mathbf{F}_{\leq x}\varphi \mid \mathbf{G}_{\leq y}\varphi$$

with $x \in \mathcal{X}$, $y \in \mathcal{Y}$ ($\mathcal{X} \cap \mathcal{Y} = \emptyset$).

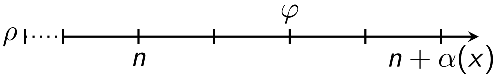
Parametric LTL

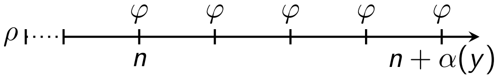
Alur et al. '99: add parameterized operators to LTL

$$\varphi ::= p \mid \neg p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \mathbf{X}\varphi \mid \varphi\mathbf{U}\varphi \mid \varphi\mathbf{R}\varphi \mid \mathbf{F}_{\leq x}\varphi \mid \mathbf{G}_{\leq y}\varphi$$

with $x \in \mathcal{X}$, $y \in \mathcal{Y}$ ($\mathcal{X} \cap \mathcal{Y} = \emptyset$).

Semantics w.r.t. variable valuation $\alpha \colon \mathcal{X} \cup \mathcal{Y} \to \mathbb{N}$:

- As usual for LTL operators.
- $(\rho, n, \alpha) \models \mathbf{F}_{\leq x}\varphi$: 

- $(\rho, n, \alpha) \models \mathbf{G}_{\leq y}\varphi$: 

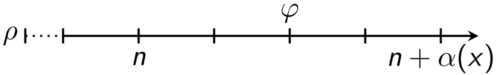
Parametric LTL

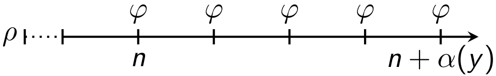
Alur et al. '99: add parameterized operators to LTL

$$\varphi ::= p \mid \neg p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \mathbf{X}\varphi \mid \varphi\mathbf{U}\varphi \mid \varphi\mathbf{R}\varphi \mid \mathbf{F}_{\leq x}\varphi \mid \mathbf{G}_{\leq y}\varphi$$

with $x \in \mathcal{X}$, $y \in \mathcal{Y}$ ($\mathcal{X} \cap \mathcal{Y} = \emptyset$).

Semantics w.r.t. variable valuation $\alpha\colon \mathcal{X} \cup \mathcal{Y} \to \mathbb{N}$:

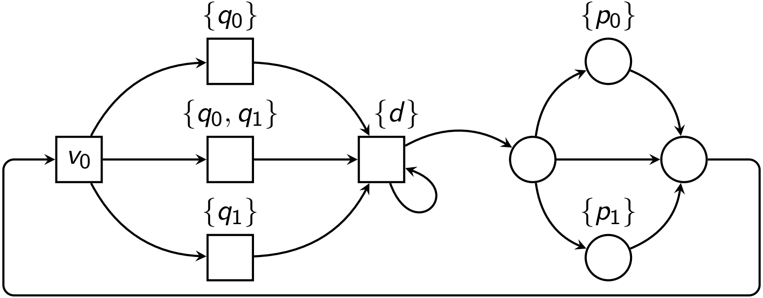
- As usual for LTL operators.
- $(\rho, n, \alpha) \models \mathbf{F}_{\leq x}\varphi$: 

- $(\rho, n, \alpha) \models \mathbf{G}_{\leq y}\varphi$: 

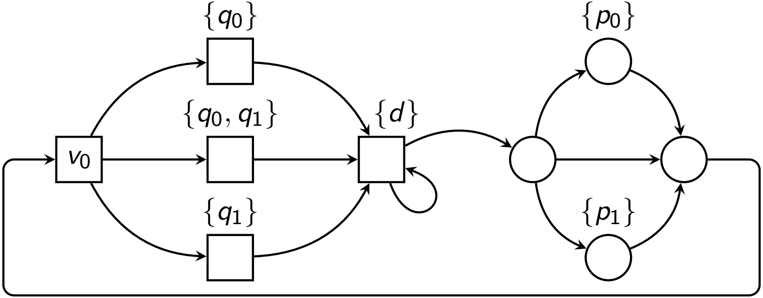
Fragments:

- $\text{PLTL}_{\mathbf{F}}$: no parameterized always operators $\mathbf{G}_{\leq y}$.
- $\text{PLTL}_{\mathbf{G}}$: no parameterized eventually operators $\mathbf{F}_{\leq x}$.

PLTL Games

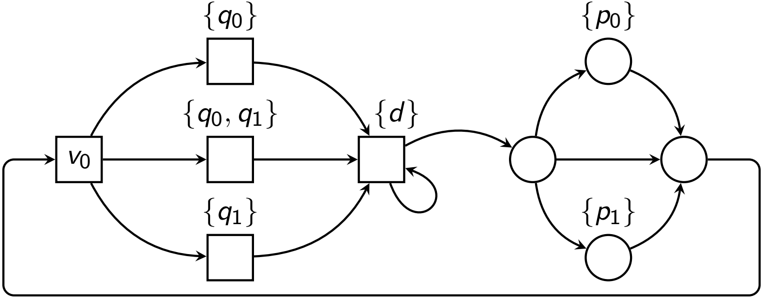


PLTL Games



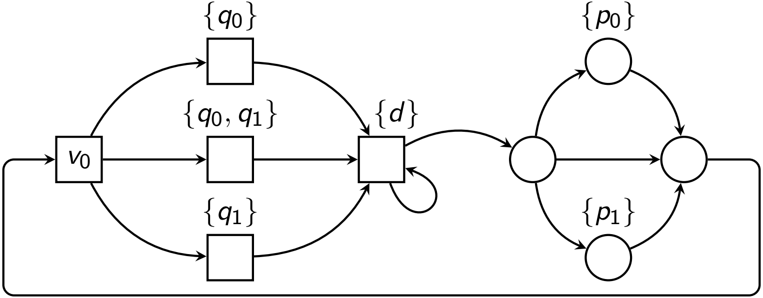
- $\varphi_1 = \mathbf{FG}d \vee \bigwedge_{i\in\{0,1\}} \mathbf{G}(q_i \rightarrow \mathbf{F}p_i)$: Player 0 wins.

PLTL Games



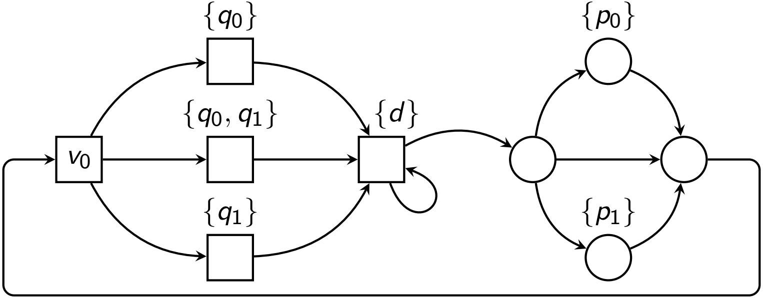
- $\varphi_1 = \mathbf{FG}d \vee \bigwedge_{i\in\{0,1\}} \mathbf{G}(q_i \rightarrow \mathbf{F}p_i)$: Player 0 wins.
- $\varphi_2 = \mathbf{FG}d \vee \bigwedge_{i\in\{0,1\}} \mathbf{G}(q_i \rightarrow \mathbf{F}_{\leq x_i}p_i)$: Player 1 wins w.r.t. every α.

PLTL Games



$\mathcal{W}_i(\mathcal{G}) = \{\alpha \mid \text{Player } i \text{ has winning strategy for } \mathcal{G} \text{ w.r.t. } \alpha\}$

PLTL Games



$\mathcal{W}_i(\mathcal{G}) = \{\alpha \mid \text{Player } i \text{ has winning strategy for } \mathcal{G} \text{ w.r.t. } \alpha\}$

Lemma (Determinacy)

$\mathcal{W}_0(\mathcal{G})$ is the complement of $\mathcal{W}_1(\mathcal{G})$.

Decision Problems

- Membership: given $\mathcal{G}$, $i \in \{0, 1\}$, and α, is $\alpha \in \mathcal{W}_i(\mathcal{G})$?
- Emptiness: given $\mathcal{G}$ and $i \in \{0, 1\}$, is $\mathcal{W}_i(\mathcal{G})$ empty?
- Finiteness: given $\mathcal{G}$ and $i \in \{0, 1\}$, is $\mathcal{W}_i(\mathcal{G})$ finite?
- Universality: given $\mathcal{G}$ and $i \in \{0, 1\}$, is $\mathcal{W}_i(\mathcal{G})$ universal?

Decision Problems

- Membership: given $\mathcal{G}$, $i \in \{0, 1\}$, and α, is $\alpha \in \mathcal{W}_i(\mathcal{G})$?
- Emptiness: given $\mathcal{G}$ and $i \in \{0, 1\}$, is $\mathcal{W}_i(\mathcal{G})$ empty?
- Finiteness: given $\mathcal{G}$ and $i \in \{0, 1\}$, is $\mathcal{W}_i(\mathcal{G})$ finite?
- Universality: given $\mathcal{G}$ and $i \in \{0, 1\}$, is $\mathcal{W}_i(\mathcal{G})$ universal?

The benchmark:

Theorem (Pnueli, Rosner '89)

Solving LTL games is $2\textsc{Exptime}$*-complete.*

Decision Problems

- Membership: given $\mathcal{G}$, $i \in \{0,1\}$, and α, is $\alpha \in \mathcal{W}_i(\mathcal{G})$?
- Emptiness: given $\mathcal{G}$ and $i \in \{0,1\}$, is $\mathcal{W}_i(\mathcal{G})$ empty?
- Finiteness: given $\mathcal{G}$ and $i \in \{0,1\}$, is $\mathcal{W}_i(\mathcal{G})$ finite?
- Universality: given $\mathcal{G}$ and $i \in \{0,1\}$, is $\mathcal{W}_i(\mathcal{G})$ universal?

The benchmark:

Theorem (Pnueli, Rosner '89)

Solving LTL games is $2\textsc{Exptime}$*-complete.*

Adding parameterized operators does not increase complexity:

Theorem (Z. '11)

All four decision problems are $2\textsc{Exptime}$*-complete.*

Proof Sketch (Emptiness)

1. Replacing $\mathbf{G}_{\leq y}\psi$ by ψ preserves emptiness (monotonicity).
2. Apply alternating color technique **(Kupferman et al. '06)**:
   - Add new proposition p and replace every $\mathbf{F}_{\leq x}\psi$ by

     $$(p \rightarrow p\mathbf{U}(\neg p\mathbf{U}\psi)) \wedge (\neg p \rightarrow \neg p\mathbf{U}(p\mathbf{U}\psi))$$

     (ψ satisfied within one color change), obtain $c(\varphi)$.

Proof Sketch (Emptiness)

1. Replacing $\mathbf{G}_{\leq y}\psi$ by ψ preserves emptiness (monotonicity).
2. Apply alternating color technique **(Kupferman et al. '06)**:
   - Add new proposition p and replace every $\mathbf{F}_{\leq x}\psi$ by

$$(p \rightarrow p\mathbf{U}(\neg p\mathbf{U}\psi)) \wedge (\neg p \rightarrow \neg p\mathbf{U}(p\mathbf{U}\psi))$$

(ψ satisfied within one color change), obtain $c(\varphi)$.

Lemma

φ and $c(\varphi)$ "equivalent" on traces where distance between color changes is bounded.

Proof Sketch (Emptiness)

1. Replacing $\mathbf{G}_{\leq y}\psi$ by ψ preserves emptiness (monotonicity).
2. Apply alternating color technique **(Kupferman et al. '06)**:
   - Add new proposition p and replace every $\mathbf{F}_{\leq x}\psi$ by

$$(p \rightarrow p\mathbf{U}(\neg p\mathbf{U}\psi)) \wedge (\neg p \rightarrow \neg p\mathbf{U}(p\mathbf{U}\psi))$$

(ψ satisfied within one color change), obtain $c(\varphi)$.

Lemma

φ and $c(\varphi)$ "equivalent" on traces where distance between color changes is bounded.

3. Emptiness for game with condition φ equivalent to Player 0 winning LTL game with condition $c(\varphi) \wedge \mathbf{GF}p \wedge \mathbf{GF}\neg p$, as finite state strategies bound distance between color changes.
4. Yields doubly-exponential upper bound.

Optimization Problems

For $\mathrm{PLTL}_{\mathbf{F}}$ and $\mathrm{PLTL}_{\mathbf{G}}$ winning conditions, synthesis is an optimization problem:

Optimization Problems

For $\text{PLTL}_{\mathbf{F}}$ and $\text{PLTL}_{\mathbf{G}}$ winning conditions, synthesis is an optimization problem:

Theorem (Z. '11)

Let $\mathcal{G}_{\mathbf{F}}$ be a $PLTL_{\mathbf{F}}$ game with winning condition $\varphi_{\mathbf{F}}$ and let $\mathcal{G}_{\mathbf{G}}$ be a $PLTL_{\mathbf{G}}$ game with winning condition $\varphi_{\mathbf{G}}$. The following values (and winning strategies realizing them) can be computed in triply-exponential time.

1. $\min_{\alpha \in \mathcal{W}_0(\mathcal{G}_{\mathbf{F}})} \min_{x \in \mathrm{var}(\varphi_{\mathbf{F}})} \alpha(x)$.

Optimization Problems

For $\text{PLTL}_{\mathbf{F}}$ and $\text{PLTL}_{\mathbf{G}}$ winning conditions, synthesis is an optimization problem:

Theorem (Z. '11)

Let $\mathcal{G}_{\mathbf{F}}$ be a $\text{PLTL}_{\mathbf{F}}$ game with winning condition $\varphi_{\mathbf{F}}$ and let $\mathcal{G}_{\mathbf{G}}$ be a $\text{PLTL}_{\mathbf{G}}$ game with winning condition $\varphi_{\mathbf{G}}$. The following values (and winning strategies realizing them) can be computed in triply-exponential time.

1. $\min_{\alpha \in \mathcal{W}_0(\mathcal{G}_{\mathbf{F}})} \min_{x \in \mathrm{var}(\varphi_{\mathbf{F}})} \alpha(x)$.
2. $\min_{\alpha \in \mathcal{W}_0(\mathcal{G}_{\mathbf{F}})} \max_{x \in \mathrm{var}(\varphi_{\mathbf{F}})} \alpha(x)$.

Optimization Problems

For $\text{PLTL}_{\mathbf{F}}$ and $\text{PLTL}_{\mathbf{G}}$ winning conditions, synthesis is an optimization problem:

Theorem (Z. '11)

Let $\mathcal{G}_{\mathbf{F}}$ be a $\text{PLTL}_{\mathbf{F}}$ game with winning condition $\varphi_{\mathbf{F}}$ and let $\mathcal{G}_{\mathbf{G}}$ be a $\text{PLTL}_{\mathbf{G}}$ game with winning condition $\varphi_{\mathbf{G}}$. The following values (and winning strategies realizing them) can be computed in triply-exponential time.

1. $\min_{\alpha \in \mathcal{W}_0(\mathcal{G}_{\mathbf{F}})} \min_{x \in \mathrm{var}(\varphi_{\mathbf{F}})} \alpha(x)$.
2. $\min_{\alpha \in \mathcal{W}_0(\mathcal{G}_{\mathbf{F}})} \max_{x \in \mathrm{var}(\varphi_{\mathbf{F}})} \alpha(x)$.
3. $\max_{\alpha \in \mathcal{W}_0(\mathcal{G}_{\mathbf{G}})} \max_{y \in \mathrm{var}(\varphi_{\mathbf{G}})} \alpha(y)$.
4. $\max_{\alpha \in \mathcal{W}_0(\mathcal{G}_{\mathbf{G}})} \min_{y \in \mathrm{var}(\varphi_{\mathbf{G}})} \alpha(y)$.

Proof Sketch

- $\min_{\alpha \in \mathcal{W}_0(\mathcal{G}_{\mathsf{F}})} \max_{x \in \mathrm{var}(\varphi_{\mathsf{F}})} \alpha(x)$ for $\mathrm{PLTL}_{\mathsf{F}}$-formula φ_{F}.

1. Replacing every variable by z preserves optimum (monotonicity).
2. Doubly-exponential upper bound on optimum.
3. Models of φ w.r.t. α recognized by deterministic parity automaton of triply-exponential size, provided $\alpha(z)$ is at most doubly-exponential.
4. Thus, $\alpha \in \mathcal{W}_0(\mathcal{G}_{\mathsf{F}})$ can be decided in triply-exponential time.
5. Run binary search over doubly-exponential search space.

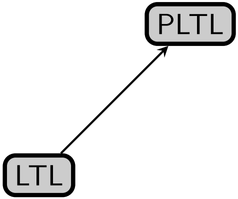
Proof Sketch

- $\min_{\alpha \in \mathcal{W}_0(\mathcal{G}_{\mathbf{F}})} \max_{x \in \mathrm{var}(\varphi_{\mathbf{F}})} \alpha(x)$ for $\mathrm{PLTL}_{\mathbf{F}}$-formula $\varphi_{\mathbf{F}}$.

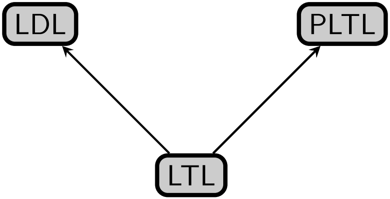
1. Replacing every variable by z preserves optimum (monotonicity).
2. Doubly-exponential upper bound on optimum.
3. Models of φ w.r.t. α recognized by deterministic parity automaton of triply-exponential size, provided $\alpha(z)$ is at most doubly-exponential.
4. Thus, $\alpha \in \mathcal{W}_0(\mathcal{G}_{\mathbf{F}})$ can be decided in triply-exponential time.
5. Run binary search over doubly-exponential search space.

Note:
Doubly-exponential lower bound on optimum rules out doubly-exponential running time for this algorithm.

Overview



Overview



Linear Dynamic Logic

Vardi '11: Another extension of LTL expressing exactly the ω-regular languages: use PDL-like operators

$$\varphi ::= p \mid \neg p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \langle r \rangle \varphi \mid [r]\varphi$$

$$r ::= \phi \mid \varphi? \mid r + r \mid r \,;\, r \mid r^*$$

where ϕ ranges over boolean formulas over atomic propositions.

Linear Dynamic Logic

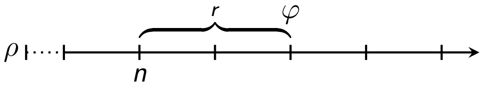
Vardi '11: Another extension of LTL expressing exactly the ω-regular languages: use PDL-like operators

$$\varphi ::= p \mid \neg p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \langle r \rangle \varphi \mid [r]\varphi$$
$$r ::= \phi \mid \varphi? \mid r + r \mid r \,;\, r \mid r^*$$

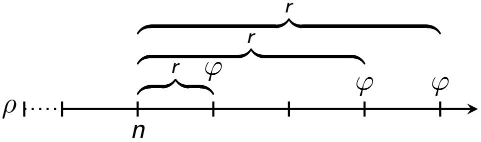
where ϕ ranges over boolean formulas over atomic propositions.

Semantics:

- $(\rho, n, \alpha) \models \langle r \rangle \varphi$:



- $(\rho, n, \alpha) \models [r]\varphi$:



Results

Theorem (Vardi '11)

LDL can be translated into linearly-sized alternating automata.

Results

Theorem (Vardi '11)

LDL can be translated into linearly-sized alternating automata.

Corollary

1. *LDL model checking is* PSPACE*-complete.*
2. *Solving games with LDL winning conditions is* 2EXPTIME*-complete.*

Results

Theorem (Vardi '11)

LDL can be translated into linearly-sized alternating automata.

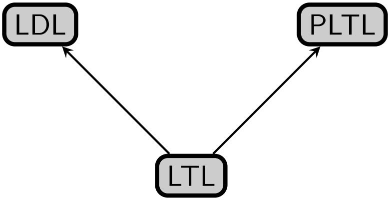
Corollary

1. *LDL model checking is* PSPACE*-complete.*
2. *Solving games with LDL winning conditions is* 2EXPTIME*-complete.*

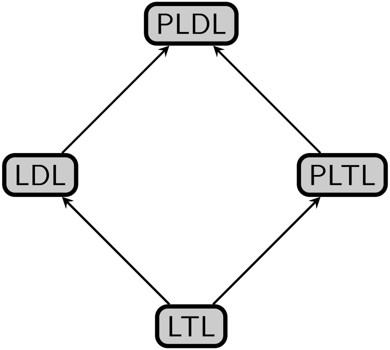
Theorem (Vardi '11)

LDL defines exactly the ω-regular languages.

Overview



Overview



Parametric LDL

Faymonville, Z. '14: add parameterized operators to LDL.

$$\varphi ::= p \mid \neg p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \langle r \rangle \varphi \mid [r]\varphi \mid \langle r \rangle_{\leq x} \varphi \mid [r]_{\leq y} \varphi$$

$$r ::= \phi \mid \varphi? \mid r + r \mid r\,;r \mid r^*$$

Parametric LDL

Faymonville, Z. '14: add parameterized operators to LDL.

$$\varphi ::= p \mid \neg p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \langle r \rangle \varphi \mid [r]\varphi \mid \langle r \rangle_{\leq x}\varphi \mid [r]_{\leq y}\varphi$$
$$r ::= \phi \mid \varphi? \mid r + r \mid r \,;\, r \mid r^*$$

We are interested in same decision problems as for PLTL

- Membership: given $\mathcal{G}$, $i \in \{0,1\}$, and α, is $\alpha \in \mathcal{W}_i(\mathcal{G})$?
- Emptiness: given $\mathcal{G}$ and $i \in \{0,1\}$, is $\mathcal{W}_i(\mathcal{G})$ empty?
- Finiteness: given $\mathcal{G}$ and $i \in \{0,1\}$, is $\mathcal{W}_i(\mathcal{G})$ finite?
- Universality: given $\mathcal{G}$ and $i \in \{0,1\}$, is $\mathcal{W}_i(\mathcal{G})$ universal?

as well as the optimization problems.

The Alternating Color Technique for PLDL

1. Eliminate $[r]_{\leq y}\psi$:

Lemma

For every r there is an $\hat{r}$ such that $[r]_{\leq y}\psi$ holds for $\alpha(y) = 0$ if and only if $[\hat{r}]\psi$ holds.

The Alternating Color Technique for PLDL

1. Eliminate $[r]_{\leq y}\psi$:

Lemma

For every r there is an $\hat{r}$ such that $[r]_{\leq y}\psi$ holds for $\alpha(y) = 0$ if and only if $[\hat{r}]\psi$ holds.

2. Eliminate $\langle r\rangle_{\leq x}\psi$ using alternating color technique:
   - Need to match r and $p^*(\neg p)^* + (\neg p)^* p^*$.
   - Introduce color change aware operators: $\langle r\rangle_{\mathrm{cc}}$ only takes into account matches of r within at most one color change.
   - Thus, replace $\langle r\rangle_{\leq x}\psi$ by $\langle r\rangle_{\mathrm{cc}}\psi$, obtain $c(\varphi)$.

The Alternating Color Technique for PLDL

1. Eliminate $[r]_{\leq y}\psi$:

Lemma

For every r there is an $\hat{r}$ such that $[r]_{\leq y}\psi$ holds for $\alpha(y) = 0$ if and only if $[\hat{r}]\psi$ holds.

2. Eliminate $\langle r\rangle_{\leq x}\psi$ using alternating color technique:
   - Need to match r and $p^*(\neg p)^* + (\neg p)^* p^*$.
   - Introduce color change aware operators: $\langle r\rangle_{\mathrm{cc}}$ only takes into account matches of r within at most one color change.
   - Thus, replace $\langle r\rangle_{\leq x}\psi$ by $\langle r\rangle_{\mathrm{cc}}\psi$, obtain $c(\varphi)$.

Lemma

φ and $c(\varphi)$ "equivalent" on traces where distance between color changes is bounded.

Translating LDL_{cc} into Automata

Theorem (Faymonville, Z. '14)

LDL_{cc}-formulas can be translated into linearly-sized alternating automata.

Proof Sketch:

Bottom up construction:

- Atomic formulas, conjunction, and disjunction straightforward.

Translating LDL_{cc} into Automata

Theorem (Faymonville, Z. '14)

LDL_{cc}-formulas can be translated into linearly-sized alternating automata.

Proof Sketch:

Bottom up construction:

- Atomic formulas, conjunction, and disjunction straightforward.
- $\langle r \rangle \psi$: construct NFA $\mathfrak{A}_r$ for r and alternating automaton $\mathfrak{A}_\psi$ for ψ, connect final states of $\mathfrak{A}_r$ with initial state of $\mathfrak{A}_\psi$.

Translating LDL_{cc} into Automata

Theorem (Faymonville, Z. '14)

LDL_{cc}-formulas can be translated into linearly-sized alternating automata.

Proof Sketch:

Bottom up construction:

- Atomic formulas, conjunction, and disjunction straightforward.
- $\langle r \rangle \psi$: construct NFA $\mathfrak{A}_r$ for r and alternating automaton $\mathfrak{A}_\psi$ for ψ, connect final states of $\mathfrak{A}_r$ with initial state of $\mathfrak{A}_\psi$.
- $[r]\psi$: as for $\langle r \rangle \psi$, but make all states of $\mathfrak{A}_r$ universal to test for all matches of r.

Translating $\mathrm{LDL}_{\mathrm{cc}}$ into Automata

Theorem (Faymonville, Z. '14)

LDL_{cc}-formulas can be translated into linearly-sized alternating automata.

Proof Sketch:

Bottom up construction:

- Atomic formulas, conjunction, and disjunction straightforward.
- $\langle r\rangle\psi$: construct NFA $\mathfrak{A}_r$ for r and alternating automaton $\mathfrak{A}_\psi$ for ψ, connect final states of $\mathfrak{A}_r$ with initial state of $\mathfrak{A}_\psi$.
- $[r]\psi$: as for $\langle r\rangle\psi$, but make all states of $\mathfrak{A}_r$ universal to test for all matches of r.
- $\langle r\rangle_{\mathrm{cc}}\psi$: as for $\langle r\rangle\psi$, but take intersection of $\mathfrak{A}_r$ and automaton checking for at most one color change.

Results for LDL

Theorem (Faymonville, Z. '14)

PLDL model checking is PSPACE*-complete.*

Results for LDL

Theorem (Faymonville, Z. '14)

PLDL model checking is PSPACE*-complete.*

Theorem (Faymonville, Z. '14)

Solving games with PLDL winning conditions is 2EXPTIME*-complete.*

Results for LDL

Theorem (Faymonville, Z. '14)

PLDL model checking is PSPACE*-complete.*

Theorem (Faymonville, Z. '14)

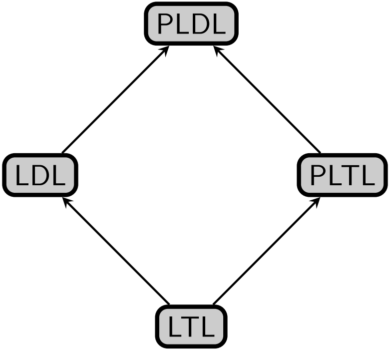
Solving games with PLDL winning conditions is 2EXPTIME*-complete.*

Theorem (Faymonville, Z. '14)

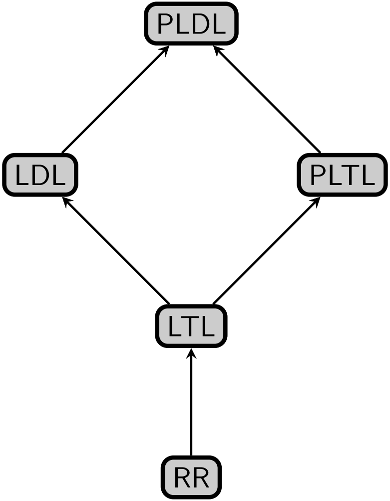
PLDL optimization problems are solvable in

- *polynomial space for model checking, and*
- *triply-exponential time for games.*

Overview



Overview



Request-Response Conditions

For propositions q_j (requests) and p_j (responses)

$$\varphi = \bigwedge_{j=1}^{k} \mathbf{G}(q_j \rightarrow \mathbf{F} p_j)$$

Request-Response Conditions

For propositions q_j (requests) and p_j (responses)

$$\varphi = \bigwedge_{j=1}^{k} \mathbf{G}(q_j \rightarrow \mathbf{F} p_j)$$

From now on:

- RR game $(\mathcal{A}, (Q_j, P_j)_{j=1,\ldots,k})$.
- Player 0 wins if every request is answered by corresponding response, i.e., if φ is satisfied.

Request-Response Conditions

For propositions q_j (requests) and p_j (responses)

$$\varphi = \bigwedge_{j=1}^{k} \mathbf{G}(q_j \rightarrow \mathbf{F} p_j)$$

From now on:

- RR game $(\mathcal{A}, (Q_j, P_j)_{j=1,\dots,k})$.
- Player 0 wins if every request is answered by corresponding response, i.e., if φ is satisfied.

Theorem (Wallmeier, Hütten, Thomas '03)

RR games can be reduced to Büchi games of size $|\mathcal{A}|k2^{k+1}$.

Request-Response Conditions

For propositions q_j (requests) and p_j (responses)

$$\varphi = \bigwedge_{j=1}^{k} \mathbf{G}(q_j \rightarrow \mathbf{F} p_j)$$

From now on:

- RR game $(\mathcal{A}, (Q_j, P_j)_{j=1,\ldots,k})$.
- Player 0 wins if every request is answered by corresponding response, i.e., if φ is satisfied.

Theorem (Wallmeier, Hütten, Thomas '03)

RR games can be reduced to Büchi games of size $|\mathcal{A}|k2^{k+1}$.

Corollary

- *Finite-state winning strategies of size $k2^{k+1}$ for both players.*
- *Solvable in* EXPTIME.

Waiting Times

- $\mathrm{wt}_j(\varepsilon) = 0$, and

$$\mathrm{wt}_j(wv) = \begin{cases} 0 & \text{if } \mathrm{wt}_j(w) = 0 \text{ and } v \notin Q_j \setminus P_j, \\ & \\ & \end{cases}$$

Waiting Times

- $\mathrm{wt}_j(\varepsilon) = 0$, and

$$\mathrm{wt}_j(wv) = \begin{cases} 0 & \text{if } \mathrm{wt}_j(w) = 0 \text{ and } v \notin Q_j \setminus P_j, \\ 1 & \text{if } \mathrm{wt}_j(w) = 0 \text{ and } v \in Q_j \setminus P_j, \end{cases}$$

Waiting Times

- $\mathrm{wt}_j(\varepsilon) = 0$, and

$$\mathrm{wt}_j(wv) = \begin{cases} 0 & \text{if } \mathrm{wt}_j(w) = 0 \text{ and } v \notin Q_j \setminus P_j, \\ 1 & \text{if } \mathrm{wt}_j(w) = 0 \text{ and } v \in Q_j \setminus P_j, \\ 0 & \text{if } \mathrm{wt}_j(w) > 0 \text{ and } v \in P_j, \\ & \end{cases}$$

Waiting Times

- $\mathrm{wt}_j(\varepsilon) = 0$, and

$$\mathrm{wt}_j(wv) = \begin{cases} 0 & \text{if } \mathrm{wt}_j(w) = 0 \text{ and } v \notin Q_j \setminus P_j, \\ 1 & \text{if } \mathrm{wt}_j(w) = 0 \text{ and } v \in Q_j \setminus P_j, \\ 0 & \text{if } \mathrm{wt}_j(w) > 0 \text{ and } v \in P_j, \\ \mathrm{wt}_j(w) + 1 & \text{if } \mathrm{wt}_j(w) > 0 \text{ and } v \notin P_j. \end{cases}$$

Waiting Times

- $\mathrm{wt}_j(\varepsilon) = 0$, and

$$\mathrm{wt}_j(wv) = \begin{cases} 0 & \text{if } \mathrm{wt}_j(w) = 0 \text{ and } v \notin Q_j \setminus P_j, \\ 1 & \text{if } \mathrm{wt}_j(w) = 0 \text{ and } v \in Q_j \setminus P_j, \\ 0 & \text{if } \mathrm{wt}_j(w) > 0 \text{ and } v \in P_j, \\ \mathrm{wt}_j(w) + 1 & \text{if } \mathrm{wt}_j(w) > 0 \text{ and } v \notin P_j. \end{cases}$$

- $\mathrm{val}(\rho) = \limsup_{n\to\infty} \frac{1}{n} \sum_{\ell=0}^{n-1} \sum_{j=1,\ldots,k} \mathrm{wt}_j(\rho_0 \cdots \rho_\ell)$

Waiting Times

- $\mathrm{wt}_j(\varepsilon) = 0$, and

$$\mathrm{wt}_j(wv) = \begin{cases} 0 & \text{if } \mathrm{wt}_j(w) = 0 \text{ and } v \notin Q_j \setminus P_j, \\ 1 & \text{if } \mathrm{wt}_j(w) = 0 \text{ and } v \in Q_j \setminus P_j, \\ 0 & \text{if } \mathrm{wt}_j(w) > 0 \text{ and } v \in P_j, \\ \mathrm{wt}_j(w) + 1 & \text{if } \mathrm{wt}_j(w) > 0 \text{ and } v \notin P_j. \end{cases}$$

- $\mathrm{val}(\rho) = \limsup_{n\to\infty} \frac{1}{n} \sum_{\ell=0}^{n-1} \sum_{j=1,\ldots,k} \mathrm{wt}_j(\rho_0 \cdots \rho_\ell)$
- $\mathrm{val}(\sigma, v) = \sup_{\rho \in \mathrm{Beh}(v,\sigma)} \mathrm{val}(\rho)$

Waiting Times

- $\mathrm{wt}_j(\varepsilon) = 0$, and

$$\mathrm{wt}_j(wv) = \begin{cases} 0 & \text{if } \mathrm{wt}_j(w) = 0 \text{ and } v \notin Q_j \setminus P_j, \\ 1 & \text{if } \mathrm{wt}_j(w) = 0 \text{ and } v \in Q_j \setminus P_j, \\ 0 & \text{if } \mathrm{wt}_j(w) > 0 \text{ and } v \in P_j, \\ \mathrm{wt}_j(w) + 1 & \text{if } \mathrm{wt}_j(w) > 0 \text{ and } v \notin P_j. \end{cases}$$

- $\mathrm{val}(\rho) = \limsup_{n\to\infty} \frac{1}{n} \sum_{\ell=0}^{n-1} \sum_{j=1,\ldots,k} \mathrm{wt}_j(\rho_0 \cdots \rho_\ell)$
- $\mathrm{val}(\sigma, v) = \sup_{\rho \in \mathrm{Beh}(v,\sigma)} \mathrm{val}(\rho)$

Goal:
Prove that optimal winning strategies exist and are computable.

Main Theorem

Theorem

Optimal strategies for RR games exist, are effectively computable, and finite-state.

Main Theorem

Theorem

Optimal strategies for RR games exist, are effectively computable, and finite-state.

Proof strategy:

1. Strategies of *small* value can be turned into strategies with bounded waiting times without increasing the value.

Main Theorem

Theorem

Optimal strategies for RR games exist, are effectively computable, and finite-state.

Proof strategy:

1. Strategies of *small* value can be turned into strategies with bounded waiting times without increasing the value.
   - This applies to optimal strategies as well.
   - Makes the search space for optimal strategies finite.
   - Involves removing parts of plays with large waiting times.

Main Theorem

Theorem

Optimal strategies for RR games exist, are effectively computable, and finite-state.

Proof strategy:

1. Strategies of *small* value can be turned into strategies with bounded waiting times without increasing the value.
   - This applies to optimal strategies as well.
   - Makes the search space for optimal strategies finite.
   - Involves removing parts of plays with large waiting times.
2. Expand arena by keeping track of waiting time vectors up to bound from 1.). RR-values equal to mean-payoff condition.

Main Theorem

Theorem

Optimal strategies for RR games exist, are effectively computable, and finite-state.

Proof strategy:

1. Strategies of *small* value can be turned into strategies with bounded waiting times without increasing the value.
   - This applies to optimal strategies as well.
   - Makes the search space for optimal strategies finite.
   - Involves removing parts of plays with large waiting times.
2. Expand arena by keeping track of waiting time vectors up to bound from 1.). RR-values equal to mean-payoff condition.
   - Optimal strategy for mean-payoff yields optimal strategy for RR game.

Dickson's Lemma

Dickson pair: $((x_1, \ldots, x_k), (y_1, \ldots, y_k)) \in \mathbb{N}^k$ s.t. $x_j \leq y_j$ for all j.

Lemma (Dickson '13)

$(\mathbb{N}^k, \leq)$ *is a WQO, i.e., every infinite sequence has dickson pair.*

Dickson's Lemma

Dickson pair: $((x_1, \ldots, x_k), (y_1, \ldots, y_k)) \in \mathbb{N}^k$ s.t. $x_j \leq y_j$ for all j.

Lemma (Dickson '13)

$(\mathbb{N}^k, \leq)$ is a WQO, i.e., every infinite sequence has dickson pair.

However, Dickson's Lemma does not give any bound on length of infixes without dickson pairs (as there is none for $\mathbb{N}^k$).

Dickson's Lemma

Dickson pair: $((x_1, \dots, x_k), (y_1, \dots, y_k)) \in \mathbb{N}^k$ s.t. $x_j \leq y_j$ for all j.

Lemma (Dickson '13)

$(\mathbb{N}^k, \leq)$ is a WQO, i.e., every infinite sequence has dickson pair.

However, Dickson's Lemma does not give any bound on length of infixes without dickson pairs (as there is none for $\mathbb{N}^k$).

Waiting time vectors are special:

- either increment, or
- reset to zero.

Lemma

There is a function $b(|\mathcal{A}|, k) \in \mathcal{O}(2^{2^{|\mathcal{A}| \cdot k + 2}})$ such that every play infix of length $b(|\mathcal{A}|, k)$ has a dickson pair.

Bounding the Waiting Times

We have σ with $\mathrm{val}(\sigma, v) \leq \sum_{j=1,\ldots,k} |\mathcal{A}| k 2^k =: b_{\mathcal{G}}$ for all $v \in W_0(\mathcal{G})$.

Bounding the Waiting Times

We have σ with $\mathrm{val}(\sigma, v) \leq \sum_{j=1,\ldots,k} |\mathcal{A}|k2^k =: b_{\mathcal{G}}$ for all $v \in W_0(\mathcal{G})$.

Lemma

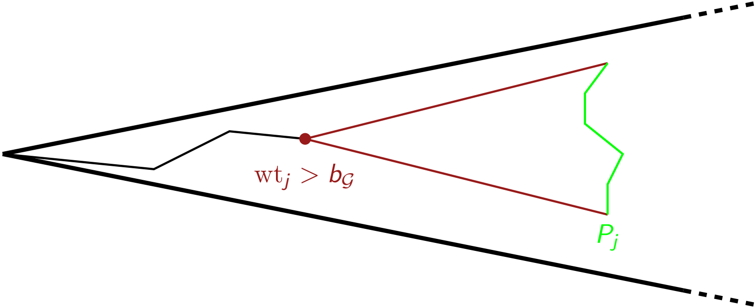
Let σ be s.t. $\mathrm{val}(\sigma, v) \leq b_{\mathcal{G}}$ for all $v \in W_0(\mathcal{G})$. There is σ' with $\mathrm{val}(\sigma', v) \leq \mathrm{val}(\sigma, v)$ for all v that uniformly bounds the waiting times for every condition j by $b_{\mathcal{G}} + b(|\mathcal{A}|, k-1)$.

Bounding the Waiting Times

We have σ with $\mathrm{val}(\sigma, v) \leq \sum_{j=1,\ldots,k} |\mathcal{A}| k 2^k =: b_{\mathcal{G}}$ for all $v \in W_0(\mathcal{G})$.

Lemma

Let σ be s.t. $\mathrm{val}(\sigma, v) \leq b_{\mathcal{G}}$ for all $v \in W_0(\mathcal{G})$. There is σ' with $\mathrm{val}(\sigma', v) \leq \mathrm{val}(\sigma, v)$ for all v that uniformly bounds the waiting times for every condition j by $b_{\mathcal{G}} + b(|\mathcal{A}|, k-1)$.

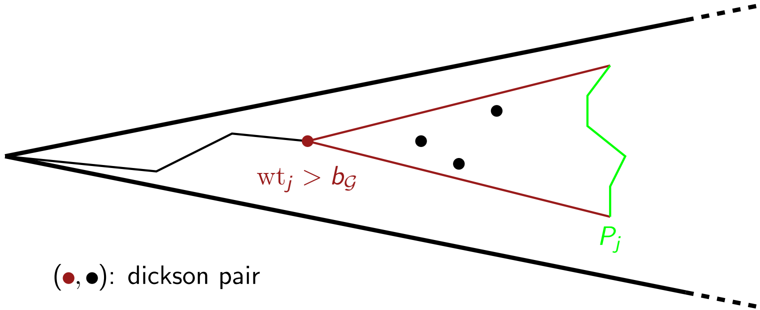


Bounding the Waiting Times

We have σ with $\mathrm{val}(\sigma, v) \leq \sum_{j=1,\ldots,k} |\mathcal{A}| k 2^k =: b_{\mathcal{G}}$ for all $v \in W_0(\mathcal{G})$.

Lemma

Let σ be s.t. $\mathrm{val}(\sigma, v) \leq b_{\mathcal{G}}$ for all $v \in W_0(\mathcal{G})$. There is σ' with $\mathrm{val}(\sigma', v) \leq \mathrm{val}(\sigma, v)$ for all v that uniformly bounds the waiting times for every condition j by $b_{\mathcal{G}} + b(|\mathcal{A}|, k-1)$.



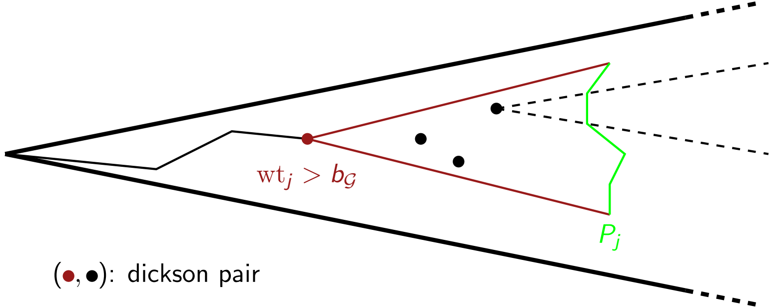
($\bullet$, $\bullet$): dickson pair

Bounding the Waiting Times

We have σ with $\mathrm{val}(\sigma, v) \leq \sum_{j=1,\ldots,k} |\mathcal{A}| k 2^k =: b_{\mathcal{G}}$ for all $v \in W_0(\mathcal{G})$.

Lemma

Let σ be s.t. $\mathrm{val}(\sigma, v) \leq b_{\mathcal{G}}$ for all $v \in W_0(\mathcal{G})$. There is σ' with $\mathrm{val}(\sigma', v) \leq \mathrm{val}(\sigma, v)$ for all v that uniformly bounds the waiting times for every condition j by $b_{\mathcal{G}} + b(|\mathcal{A}|, k-1)$.

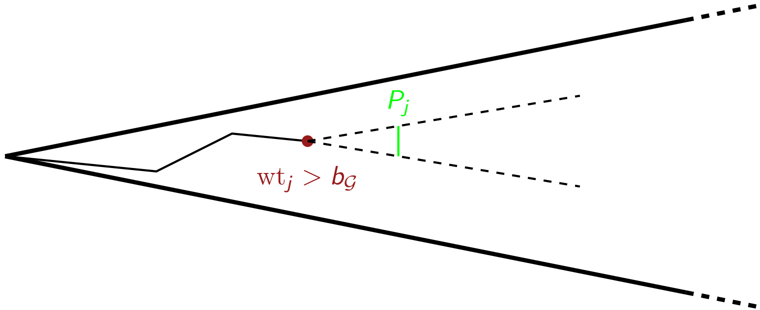


Bounding the Waiting Times

We have σ with $\mathrm{val}(\sigma, v) \leq \sum_{j=1,\dots,k} |\mathcal{A}|k2^k =: b_{\mathcal{G}}$ for all $v \in W_0(\mathcal{G})$.

Lemma

Let σ be s.t. $\mathrm{val}(\sigma, v) \leq b_{\mathcal{G}}$ for all $v \in W_0(\mathcal{G})$. There is σ' with $\mathrm{val}(\sigma', v) \leq \mathrm{val}(\sigma, v)$ for all v that uniformly bounds the waiting times for every condition j by $b_{\mathcal{G}} + b(|\mathcal{A}|, k-1)$.



From RR Games to Mean-Payoff Games

- Let $t_{\mathsf{max}_j} = \mathrm{val}_{\mathcal{G}} + b(|\mathcal{A}|, k-1)$.

From RR Games to Mean-Payoff Games

- Let $t_{\mathsf{max}_j} = \mathrm{val}_{\mathcal{G}} + b(|\mathcal{A}|, k-1)$.
- Let $\mathfrak{A}$ be DFA that keeps track of waiting vectors as long as each coordinate j is bounded by t_{max_j} (sink state $\perp$).

From RR Games to Mean-Payoff Games

- Let $t_{\mathsf{max}_j} = \mathrm{val}_{\mathcal{G}} + b(|\mathcal{A}|, k-1)$.
- Let $\mathfrak{A}$ be DFA that keeps track of waiting vectors as long as each coordinate j is bounded by t_{max_j} (sink state $\perp$).
- Take cartesian product of $\mathcal{A}$ and $\mathfrak{A}$.

From RR Games to Mean-Payoff Games

- Let $t_{\mathsf{max}_j} = \mathrm{val}_{\mathcal{G}} + b(|\mathcal{A}|, k-1)$.
- Let $\mathfrak{A}$ be DFA that keeps track of waiting vectors as long as each coordinate j is bounded by t_{max_j} (sink state $\perp$).
- Take cartesian product of $\mathcal{A}$ and $\mathfrak{A}$.
- Define w by $w((v, \perp), (v', \perp)) = 1 + \sum_{j=1,\ldots,k} t_{\mathsf{max}_j}$ and

$$w((v, (t_1, \ldots, t_k)), (v', (t'_1, \ldots, t'_k))) = \sum_{j=1,\ldots,k} t_j$$

From RR Games to Mean-Payoff Games

- Let $t_{\mathsf{max}_j} = \mathrm{val}_{\mathcal{G}} + b(|\mathcal{A}|, k-1)$.
- Let $\mathfrak{A}$ be DFA that keeps track of waiting vectors as long as each coordinate j is bounded by t_{max_j} (sink state $\perp$).
- Take cartesian product of $\mathcal{A}$ and $\mathfrak{A}$.
- Define w by $w((v, \perp), (v', \perp)) = 1 + \sum_{j=1,\ldots,k} t_{\mathsf{max}_j}$ and

$$w((v, (t_1, \ldots, t_k)), (v', (t'_1, \ldots, t'_k))) = \sum_{j=1,\ldots,k} t_j$$

- Obtain mean-payoff game $\mathcal{G}' = (\mathcal{A} \times \mathfrak{A}, w)$.

From RR Games to Mean-Payoff Games

- Let $t_{\mathsf{max}_j} = \mathrm{val}_{\mathcal{G}} + b(|\mathcal{A}|, k-1)$.
- Let $\mathfrak{A}$ be DFA that keeps track of waiting vectors as long as each coordinate j is bounded by t_{max_j} (sink state $\perp$).
- Take cartesian product of $\mathcal{A}$ and $\mathfrak{A}$.
- Define w by $w((v, \perp), (v', \perp)) = 1 + \sum_{j=1,\ldots,k} t_{\mathsf{max}_j}$ and

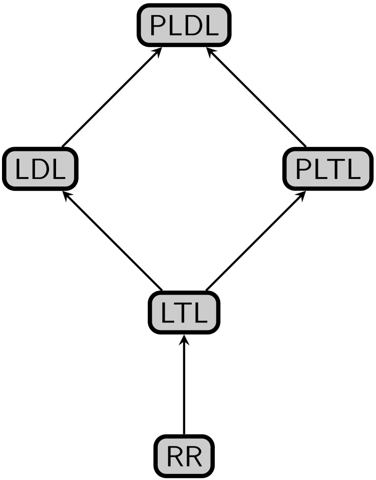
$$w((v, (t_1, \ldots, t_k)), (v', (t'_1, \ldots, t'_k))) = \sum_{j=1,\ldots,k} t_j$$

- Obtain mean-payoff game $\mathcal{G}' = (\mathcal{A} \times \mathfrak{A}, w)$.

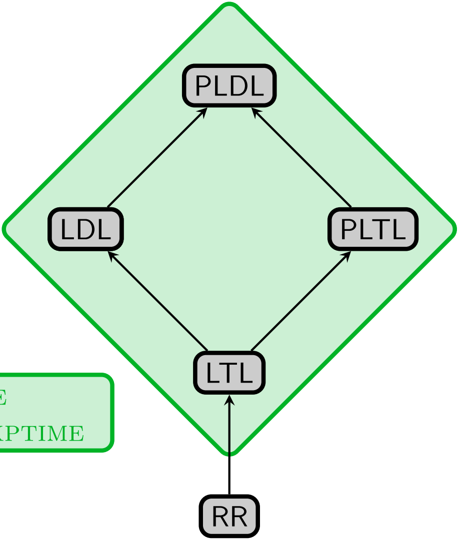
Theorem

Optimal strategy for mean-payoff game can be translated into optimal strategy for RR game.

Conclusion

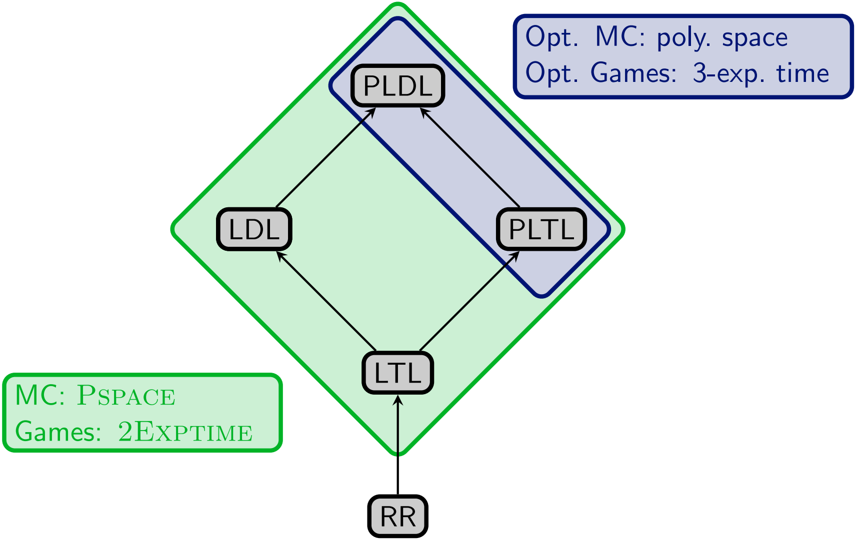


Conclusion

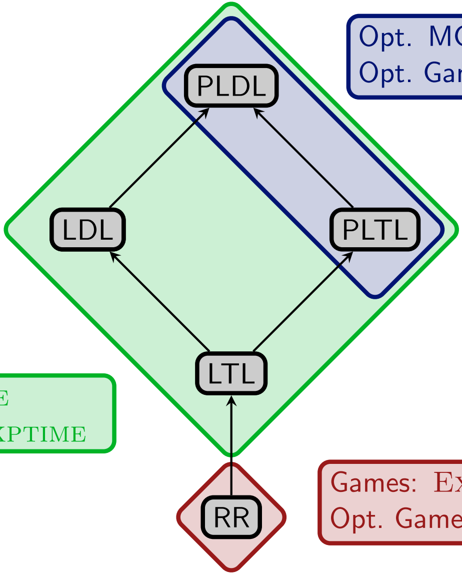


MC: PSPACE
Games: 2EXPTIME

Conclusion


Opt. MC: poly. space
Opt. Games: 3-exp. time
PLDL
LDL
PLTL
LTL
RR
MC: PSPACE
Games: 2EXPTIME

Conclusion



Opt. MC: poly. space
Opt. Games: 3-exp. time

MC: PSPACE
Games: 2EXPTIME

Games: EXPTIME
Opt. Games: 2-exp. time