Visibly Linear Dynamic Logic

Joint work with Alexander Weinert (Saarland University)

Martin Zimmermann

Saarland University

December 14th, 2016

FSTTCS 2016, Chennai, India

The Everlasting Quest for Expressiveness

Consider an arbiter granting access to a shared resource.

Requirements:

- "Every request q is eventually answered by a response p"
- "Every request q is eventually answered by a response p after an *even* number of steps"
- "There are never more responses than requests"

The Everlasting Quest for Expressiveness

Consider an arbiter granting access to a shared resource.

Requirements:

- "Every request q is eventually answered by a response p"

  Linear Temporal Logic: $\mathbf{G}(q \rightarrow \mathbf{F}\, p)$

- "Every request q is eventually answered by a response p after an *even* number of steps"

- "There are never more responses than requests"

The Everlasting Quest for Expressiveness

Consider an arbiter granting access to a shared resource.

Requirements:

- "Every request q is eventually answered by a response p"

  Linear Temporal Logic: $\mathbf{G}(q \rightarrow \mathbf{F}\, p)$

- "Every request q is eventually answered by a response p after an *even* number of steps"

  Linear Dynamic Logic: $[\mathtt{true}^*](\, q \rightarrow \langle(\mathtt{true} \cdot \mathtt{true})^*\rangle p\,)$

- "There are never more responses than requests"

The Everlasting Quest for Expressiveness

Consider an arbiter granting access to a shared resource.

Requirements:

- "Every request q is eventually answered by a response p"

  Linear Temporal Logic: $\mathbf{G}(q \rightarrow \mathbf{F}\, p)$

- "Every request q is eventually answered by a response p after an *even* number of steps"

  Linear Dynamic Logic: $[\mathtt{true}^*](\, q \rightarrow \langle(\mathtt{true} \cdot \mathtt{true})^*\rangle p\,)$

- "There are never more responses than requests"

  Expressible with pushdown automata/context-free grammars as guards $\Rightarrow$ Visibly Linear Dynamic Logic

Outline

1. **Preliminaries**

2. **Expressiveness**

3. **VLDL Verification**

4. **Discussion**

Outline

1. **Preliminaries**

2. **Expressiveness**

3. **VLDL Verification**

4. **Discussion**

Visibly Pushdown Automata

Partition input alphabet Σ into Σ_c (calls), Σ_r (returns), and Σ_ℓ (local actions).

A visibly pushdown automaton (VPA) has to

- push when processing a call,
- pop when processing a return, and
- leave the stack unchanged when processing a local action.

Stack height determined by input word $\Rightarrow$ closure under union, intersection, and complement.

Visibly Pushdown Automata

Partition input alphabet Σ into Σ_c (calls), Σ_r (returns), and Σ_ℓ (local actions).

A visibly pushdown automaton (VPA) has to

- push when processing a call,
- pop when processing a return, and
- leave the stack unchanged when processing a local action.

Stack height determined by input word $\Rightarrow$ closure under union, intersection, and complement.

Examples:

- $a^n b^n$ is a VPL, if a is a call and b a return.
- ww^R is not a VPL.

Visibly Linear Dynamic Logic (VLDL)

Syntax

$$\varphi := p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \langle\mathfrak{A}\rangle\varphi \mid [\mathfrak{A}]\varphi$$

where $p \in P$ ranges over atomic propositions and $\mathfrak{A}$ ranges over VPA's. All VPA's have the same partition of 2^P into calls, returns, and local actions.

Visibly Linear Dynamic Logic (VLDL)

Syntax

$$\varphi := p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \langle\mathfrak{A}\rangle\varphi \mid [\mathfrak{A}]\varphi$$

where $p \in P$ ranges over atomic propositions and $\mathfrak{A}$ ranges over VPA's. All VPA's have the same partition of 2^P into calls, returns, and local actions.

Semantics: $(w \in (2^P)^\omega)$

- $w \models \langle\mathfrak{A}\rangle\varphi$ if there exists an n such that $w_0 \cdots w_{n-1}$ is accepted by $\mathfrak{A}$ and $w_n w_{n+1} w_{n+2} \cdots \models \varphi$.
- $w \models [\mathfrak{A}]\varphi$ if for every n s.t. $w_0 \cdots w_{n-1}$ is accepted by $\mathfrak{A}$ we have $w_n w_{n+1} w_{n+2} \cdots \models \varphi$.

Example

"Every request q is eventually answered by a response p and there are never more responses than requests"

$$[\mathfrak{A}^*](\, q \rightarrow \langle \mathfrak{A}^* \rangle p\,) \wedge \neg \langle \mathfrak{A} \rangle \mathtt{true}$$

where

- $\mathfrak{A}^*$ accepts every word, and
- $\mathfrak{A}$ accepts those words with more responses than requests.

Both languages are visibly pushdown, if

- $\{q\}$ is a call,
- $\{p\}$ is a return, and
- $\emptyset$ and $\{p, q\}$ are local actions.

Outline

1. Preliminaries

2. **Expressiveness**

3. VLDL Verification

4. Discussion

Expressiveness

Lemma

VLDL and non-deterministic ω-VPA are expressively equivalent.

Expressiveness

Lemma

VLDL and non-deterministic ω-VPA are expressively equivalent.

Proof Idea

VLDL

non-deterministic
ω-VPA

Expressiveness

Lemma

VLDL and non-deterministic ω-VPA are expressively equivalent.

Proof Idea

VLDL

Deterministic Stair Automata

$\mathcal{O}(2^n)$

[LMS '04]

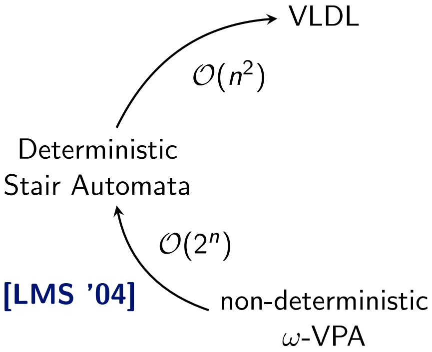
non-deterministic ω-VPA

Expressiveness

Lemma

VLDL and non-deterministic ω-VPA are expressively equivalent.

Proof Idea

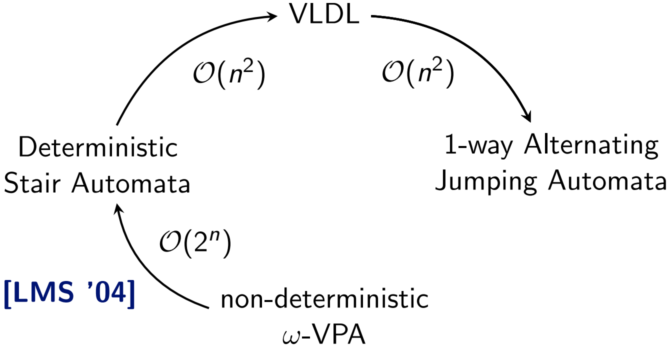


Expressiveness

Lemma

VLDL and non-deterministic ω-VPA are expressively equivalent.

Proof Idea

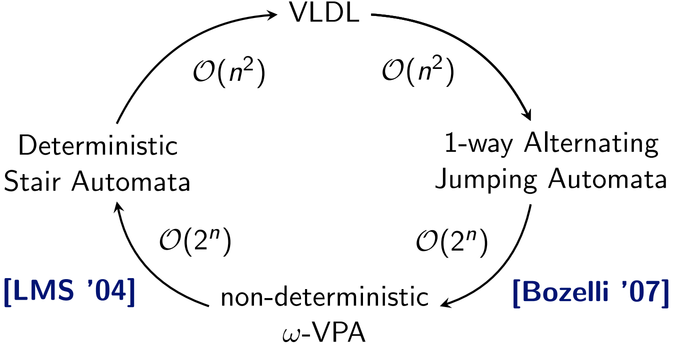


Expressiveness

Lemma

VLDL and non-deterministic ω-VPA are expressively equivalent.

Proof Idea

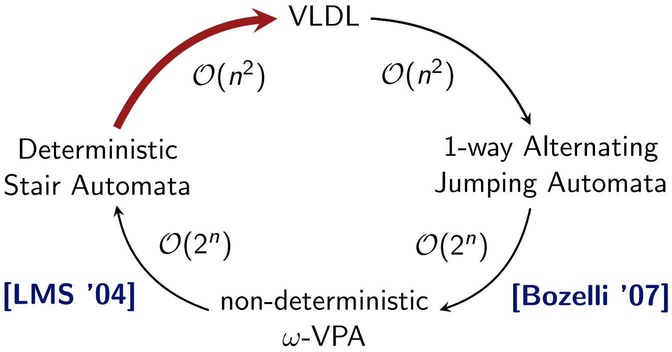


Expressiveness

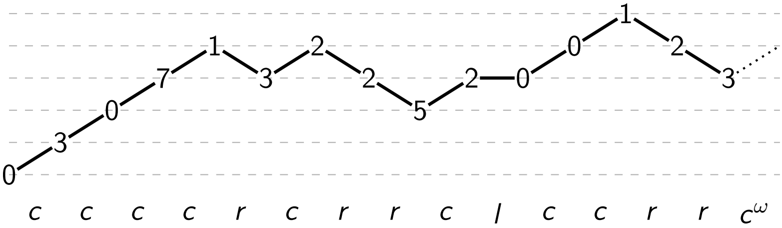
Lemma

VLDL and non-deterministic ω-VPA are expressively equivalent.

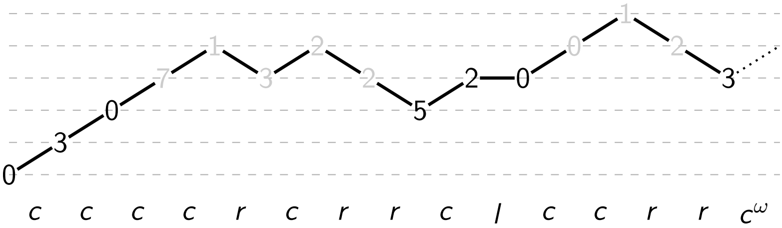
Proof Idea



From Stair Automata to VLDL

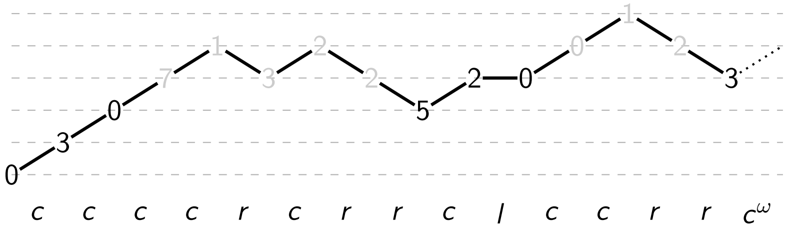


From Stair Automata to VLDL



Acceptance: maximal priority occuring at infinitely many steps even

From Stair Automata to VLDL

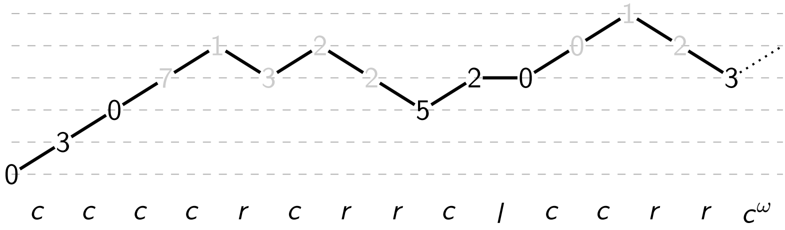


Acceptance: maximal priority occuring at infinitely many steps even

Equivalently: For some state q of even priority c there is step with state q s.t.

1. after this step, no larger priority appears at a step, and
2. for every step with state q, there is a later one with state q.

From Stair Automata to VLDL



Acceptance: maximal priority occuring at infinitely many steps even

Equivalently: For some state q of even priority c there is step with state q s.t.

1. after this step, no larger priority appears at a step, and
2. for every step with state q, there is a later one with state q.

$$\bigvee_{q \in Q_{even}} \langle {}_{q_I}\mathfrak{A}'_q \rangle \left(\bigwedge_{q' \in Q_{>\Omega(q)}} [{}_q\mathfrak{A}'_{q'}]\texttt{false} \right) \wedge [\mathfrak{A}'_q]\langle {}_q\mathfrak{A}'_q \rangle \texttt{true}$$

Outline

1. Preliminaries

2. Expressiveness

3. **VLDL Verification**

4. Discussion

Satisfiability

Theorem

VLDL Satisfiability is EXPTIME*-complete.*

Satisfiability

Theorem

VLDL Satisfiability is EXPTIME*-complete.*

Proof Sketch

- **Membership:** Construct equivalent ω-VPA and check it for emptiness.
- **Hardness:** Adapt EXPTIME-hardness proof of LTL model-checking of pushdown systems **[BEM '97]**

Model Checking

Theorem

VLDL model checking of visibly pushdown systems is EXPTIME*-complete.*

Model Checking

Theorem

VLDL model checking of visibly pushdown systems is EXPTIME*-complete.*

Proof Sketch

- **Membership:** To check $\mathcal{S} \models \varphi$, construct ω-VPA equivalent to $\neg\varphi$ and check intersection with $\mathcal{S}$ for emptiness.
- **Hardness:** Follows immediately from EXPTIME-hardness of satisfiability.

Synthesis

Theorem

Solving infinite games on visibly pushdown graphs with VLDL winning conditions is $3\textsc{ExpTime}$*-complete.*

Synthesis

Theorem

Solving infinite games on visibly pushdown graphs with VLDL winning conditions is $3\textsc{ExpTime}$-complete.

Proof Sketch

- **Membership:** To determine the winner, construct an ω-VPA that accepts the winning condition and solve the resulting game with VPA winning condition **[LMS '04]**.
- **Hardness:** Adapt $3\textsc{ExpTime}$-hardness proof of pushdown games with LTL winning condition **[LMS '04]**.

Outline

The Competitors

"If p holds true immediately after entering module m, it shall hold immediately after the corresponding return from m as well"

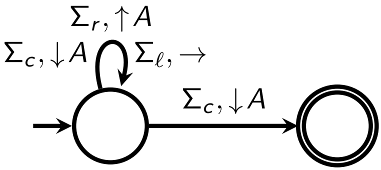
The Competitors

"If p holds true immediately after entering module m, it shall hold immediately after the corresponding return from m as well"

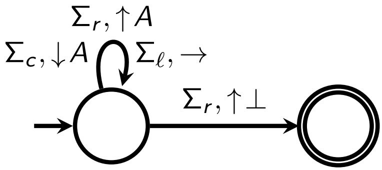
VLDL:

$$[\mathfrak{A}_c](p \rightarrow \langle \mathfrak{A}_r \rangle p)$$

with



$\mathfrak{A}_c$

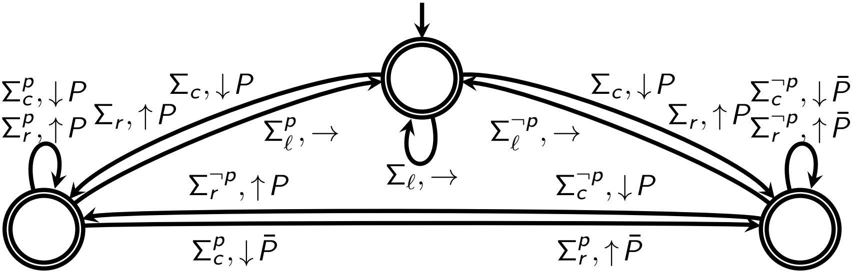


$\mathfrak{A}_r$

The Competitors

"If p holds true immediately after entering module m, it shall hold immediately after the corresponding return from m as well"

ω-VPA:



The Competitors

"If p holds true immediately after entering module m, it shall hold immediately after the corresponding return from m as well"

VLTL: [Bozzelli '14]

$$(\alpha; \texttt{true})|\alpha\rangle\texttt{false}$$

with visibly rational expression α below:

$$[(p \cup q)^* \texttt{call}_m \, [(q\Box) \cup (p\Box p)] \, \texttt{return}_m (p \cup q)^*]^{\circlearrowleft\Box} \curvearrowleft_{\Box} (p \cup q)^*$$

Conclusion

Results:

- VLDL as expressive as ω-VPA

Conclusion

Results:

- VLDL as expressive as ω-VPA
-

	validity	model-checking	infinite games
LTL	PSPACE	PSPACE	2EXPTIME
LDL	PSPACE	PSPACE	2EXPTIME

Conclusion

Results:

- VLDL as expressive as ω-VPA
-

	validity	model-checking	infinite games
LTL	PSpace	PSpace	2ExpTime
LDL	PSpace	PSpace	2ExpTime
VLDL	ExpTime	ExpTime	3ExpTime

Conclusion

Results:

- VLDL as expressive as ω-VPA
-

	validity	model-checking	infinite games
LTL	PSPACE	PSPACE	2EXPTIME
LDL	PSPACE	PSPACE	2EXPTIME
VLDL	EXPTIME	EXPTIME	3EXPTIME
VLTL	EXPTIME	EXPTIME	?

- Using (deterministic) pushdown automata as guards leads to undecidability, i.e.,

$$\langle \mathfrak{A}_1 \rangle \# \wedge \langle \mathfrak{A}_2 \rangle \# \wedge \text{“exactly one } \#\text{”}$$

is satisfiable $\Leftrightarrow L(\mathfrak{A}_1) \cap L(\mathfrak{A}_2) \neq \emptyset$.