Visibly Linear Dynamic Logic

Joint work with Alexander Weinert (Saarland University)

Martin Zimmermann

Saarland University

September 8th, 2016

Highlights Conference, Brussels, Belgium

The Everlasting Quest for Expressiveness

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

  $$\mathbf{G}(q \rightarrow \mathbf{F}\, p)$$

The Everlasting Quest for Expressiveness

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

  $$\mathbf{G}(q \rightarrow \mathbf{F}\, p)$$

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

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

The Everlasting Quest for Expressiveness

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

  $$\mathbf{G}(q \rightarrow \mathbf{F}\, p)$$

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

  $$[\texttt{true}^*](\, q \rightarrow \langle(\texttt{true} \cdot \texttt{true})^*\rangle p\,)$$

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

The Everlasting Quest for Expressiveness

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

  $$\mathbf{G}(q \rightarrow \mathbf{F}\, p)$$

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

  $$[\texttt{true}^*](\, q \rightarrow \langle(\texttt{true} \cdot \texttt{true})^*\rangle p\,)$$

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

  This can be expressed using pushdown automata/context-free grammars in the guards.

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 while the stack is non-empty (otherwise stack is unchanged), 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 while the stack is non-empty (otherwise stack is unchanged), 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

- $w \models \langle \mathfrak{A} \rangle \varphi$ if there exists an n such that $w_0 \cdots w_n$ 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. $w_0 \cdots w_n$ 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":

$$[\mathfrak{A}_{\texttt{true}}](\, q \rightarrow \langle \mathfrak{A}_{\texttt{true}} \rangle p \,) \wedge [\mathfrak{A}]\texttt{false}$$

where

- $\mathfrak{A}_{\texttt{true}}$ accepts every input, and
- $\mathfrak{A}$ accepts every input 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.

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

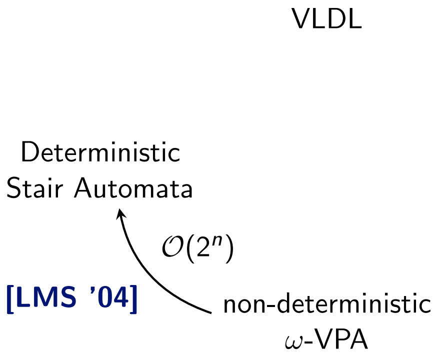
non-deterministic
ω-VPA

Expressiveness

Lemma

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

Proof Idea

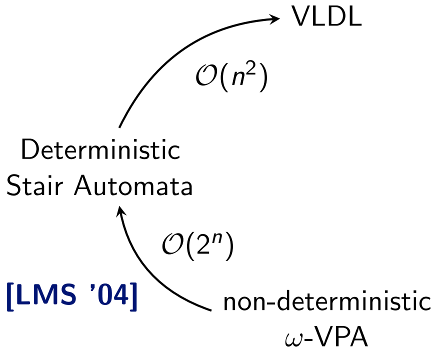


Expressiveness

Lemma

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

Proof Idea

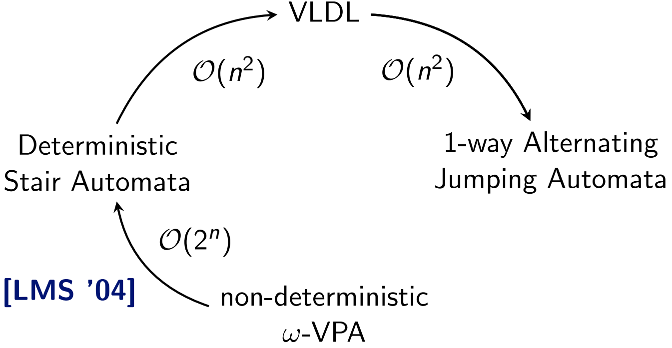


Expressiveness

Lemma

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

Proof Idea

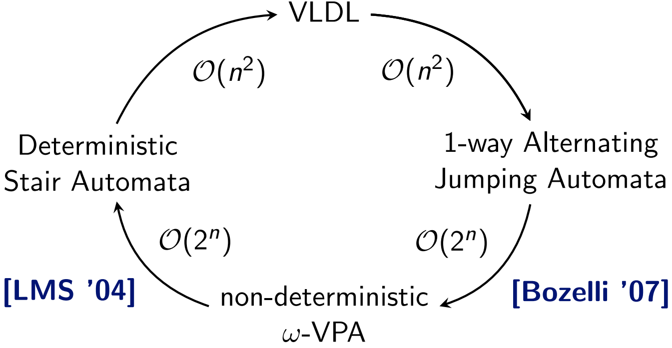


Expressiveness

Lemma

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

Proof Idea



The Competitors

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

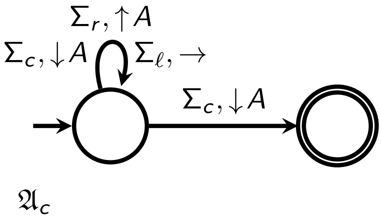
The Competitors

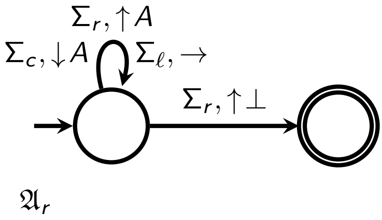
"If p holds true immediately after entering module m, it shall hold immediately after the corresponding return from m as well"

VLDL:

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

with

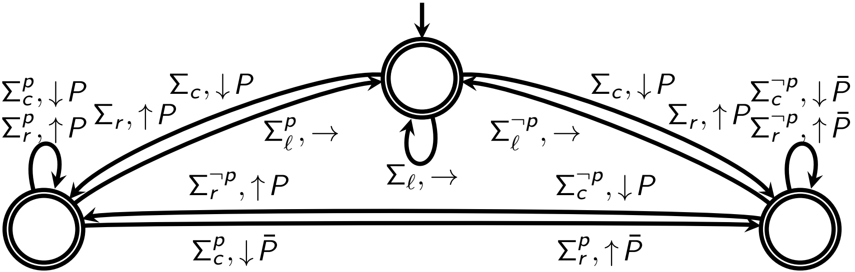




The Competitors

"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:

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

with visibly rational expression α below:

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

Our Results

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

Our Results

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

Our Results

	validity	model-checking	infinite games
LTL	PSpace	PSpace	2ExpTime
LDL	PSpace	PSpace	2ExpTime
VLDL	ExpTime	ExpTime	3ExpTime
VLTL	ExpTime	ExpTime	?
VLDL_{exp}	ExpTime	ExpTime	3ExpTime