Visibly Linear Dynamic Logic

Joint work with Alexander Weinert (Saarland University)

Martin Zimmermann
Saarland University

December 14th, 2016
FSTTCS 2016, Chennai, India
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”
Consider an arbiter granting access to a shared resource.

**Requirements:**

- “Every request $q$ is eventually answered by a response $p$”
  
  **Linear Temporal Logic:** $G(q \rightarrow F p)$

- “Every request $q$ is eventually answered by a response $p$ after an even number of steps”

- “There are never more responses than requests”
Consider an arbiter granting access to a shared resource.

**Requirements:**

- “Every request $q$ is eventually answered by a response $p$”

  **Linear Temporal Logic:** $G(q \rightarrow F p)$

- “Every request $q$ is eventually answered by a response $p$ after an even number of steps”

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

- “There are never more responses than requests”
Consider an arbiter granting access to a shared resource.

**Requirements:**

- “Every request $q$ is eventually answered by a response $p$”

  **Linear Temporal Logic:** \( G(q \rightarrow F p) \)

- “Every request $q$ is eventually answered by a response $p$ after an even number of steps”

  **Linear Dynamic Logic:** \( [\text{true}^*](q \rightarrow ((\text{true} \cdot \text{true})^*) p) \)

- “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
Partition input alphabet $\Sigma$ into $\Sigma_c$ (calls), $\Sigma_r$ (returns), and $\Sigma_\ell$ (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 $\Sigma$ into $\Sigma_c$ (calls), $\Sigma_r$ (returns), and $\Sigma_\ell$ (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.
Syntax

\[ \phi ::= p \mid \neg \phi \mid \phi \land \phi \mid \phi \lor \phi \mid \langle A \rangle \phi \mid [A] \phi \]

where \( p \in P \) ranges over atomic propositions and \( A \) 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

\[ \varphi ::= p \mid \neg \varphi \mid \varphi \land \varphi \mid \varphi \lor \varphi \mid \langle A \rangle \varphi \mid [A] \varphi \]

where \( p \in P \) ranges over atomic propositions and \( A \) ranges over VPA’s. All VPA’s have the same partition of \( 2^P \) into calls, returns, and local actions.

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

- \( w \models \langle A \rangle \varphi \) if there exists an \( n \) such that \( w_0 \cdots w_{n-1} \) is accepted by \( A \) and \( w_n w_{n+1} w_{n+2} \cdots \models \varphi \).

- \( w \models [A] \varphi \) if for every \( n \) s.t. \( w_0 \cdots w_{n-1} \) is accepted by \( A \) we have \( w_n w_{n+1} w_{n+2} \cdots \models \varphi \).
“Every request $q$ is eventually answered by a response $p$ and there are never more responses than requests”

$$[A^*](q \rightarrow \langle A^* \rangle p) \land \neg \langle A \rangle \text{true}$$

where

- $A^*$ accepts every word, and
- $A$ 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
Lemma

VLDL and non-deterministic $\omega$-VPA are expressively equivalent.
**Expressiveness**

**Lemma**

*VLDL and non-deterministic $\omega$-VPA are expressively equivalent.*

**Proof Idea**

\[ \text{VLDL} \]

\[ \text{non-deterministic} \]

\[ \text{$\omega$-VPA} \]
Expressiveness

Lemma

VLDL and non-deterministic $\omega$-VPA are expressively equivalent.

Proof Idea

VLDL

Deterministic Stair Automata

$O(2^n)$

non-deterministic $\omega$-VPA

[Bozelli '07]

$O(n^2)$

[LMS '04]
Lemma

VLDL and non-deterministic $\omega$-VPA are expressively equivalent.

Proof Idea
Lemma

VLDL and non-deterministic $\omega$-VPA are expressively equivalent.

Proof Idea

\[ O(2^n) \quad O(n^2) \quad O(n^2) \]

Deterministic Stair Automata

[\text{LMS '04}]

non-deterministic $\omega$-VPA

1-way Alternating Jumping Automata
**Expressiveness**

**Lemma**

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

**Proof Idea**

- **Deterministic Stair Automata**
  - $O(n^2)$
- **1-way Alternating Jumping Automata**
  - $O(2^n)$
- **VLDL**
  - $O(n^2)$
- **non-deterministic ω-VPA**
  - $O(n^2)$

- [LMS '04]
- [Bozelli '07]
Lemma

*VLDL and non-deterministic $\omega$-VPA are expressively equivalent.*

Proof Idea

- Deterministic Stair Automata
  - $O(2^n)$
- non-deterministic $\omega$-VPA
  - $O(2^n)$
- 1-way Alternating Jumping Automata
  - $O(n^2)$
- VLDL
  - $O(n^2)$

[Bozelli '07]

[Bozelli '07]
Acceptance: maximal priority occurring at infinitely many steps even

Equivalently:
For some state \( q \) of even priority there is step \( q \) 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 \( q \).

\[ \bigvee_{q \in Q \text{ even}} \left( \left[ q \right]_{IA'} \land \left[ \bigwedge_{q' \in Q \text{ > } \Omega(q)} \left[ A'q' \right] \right] \land \left[ A'q \right] \left[ q \right]_{IA'} \right) \]
Acceptance: maximal priority occurring at infinitely many steps even
Acceptance: maximal priority occurring 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 \( q \).
Acceptance: maximal priority occurring 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 $q$.

\[ \bigvee_{q \in Q_{\text{even}}} \langle q, A_q' \rangle \left( \bigwedge_{q' \in Q_{>\Omega(q)}} [q A_{q'}] \text{false} \right) \land [A_q] \langle q, A_q' \rangle \text{true} \]
1. Preliminaries

2. Expressiveness

3. VLDL Verification

4. Discussion


Theorem

\textit{VLDL Satisfiability is EXPTime-complete.}
Satisfiability

**Theorem**

*VLDL Satisfiability is \( \text{ExpTime}-\text{complete} \).*

**Proof Sketch**

- **Membership:** Construct equivalent \( \omega \)-VPA and check it for emptiness.
- **Hardness:** Adapt \( \text{ExpTime} \)-hardness proof of LTL model-checking of pushdown systems [BEM ’97]
Theorem

VLDL model checking of visibly pushdown systems is $\text{ExpTime}$-complete.
Model Checking

Theorem

*VLDL model checking of visibly pushdown systems is \( \text{ExpTime-complete} \).*

Proof Sketch

- **Membership:** To check \( S \models \varphi \), construct \( \omega \)-VPA equivalent to \( \neg \varphi \) and check intersection with \( S \) for emptiness.

- **Hardness:** Follows immediately from \( \text{ExpTime} \)-hardness of satisfiability.
Theorem

Solving infinite games on visibly pushdown graphs with VLDL winning conditions is $3\exp\text{Time}$-complete.
**Theorem**

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

**Proof Sketch**

- **Membership:** To determine the winner, construct an $\omega$-VPA that accepts the winning condition and solve the resulting game with VPA winning condition [LMS ’04].

- **Hardness:** Adapt $3\text{ExpTime}$-hardness proof of pushdown games with LTL winning condition [LMS ’04].
Outline

1. Preliminaries

2. Expressiveness

3. VLDL Verification

4. Discussion
"If $p$ holds true immediately after entering module $m$, it shall hold immediately after the corresponding return from $m$ as well"
“If $p$ holds true immediately after entering module $m$, it shall hold immediately after the corresponding return from $m$ as well”

**VLDL:**

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

with

\[
\begin{align*}
\Sigma_r, \uparrow A & \quad \Sigma_c, \downarrow A \\
\Sigma_c, \downarrow A & \quad \Sigma_r, \rightarrow \\
\Sigma_c, \rightarrow & \quad \Sigma_c, \rightarrow
\end{align*}
\]
"If $p$ holds true immediately after entering module $m$, it shall hold immediately after the corresponding return from $m$ as well"

$\omega$-VPA:
“If $p$ holds true immediately after entering module $m$, it shall hold immediately after the corresponding return from $m$ as well”

**VLTL: [Bozzelli ’14]**

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

with **visibly rational expression** $\alpha$ below:

$$[(p \cup q)^* \text{call}_m [(q\square) \cup (p\square p)] \text{return}_m (p \cup q)^*] \circ \lozenge \neq \lozenge (p \cup q)^*$$
Conclusion

Results:

- VLDL as expressive as $\omega$-VPA
Conclusion

Results:

- VLDL as expressive as $\omega$-VPA

<table>
<thead>
<tr>
<th>Logic</th>
<th>Validity</th>
<th>Model-Checking</th>
<th>Infinite Games</th>
</tr>
</thead>
<tbody>
<tr>
<td>LTL</td>
<td>$\text{PSPACE}$</td>
<td>$\text{PSPACE}$</td>
<td>$2\text{ExpTime}$</td>
</tr>
<tr>
<td>LDL</td>
<td>$\text{PSPACE}$</td>
<td>$\text{PSPACE}$</td>
<td>$2\text{ExpTime}$</td>
</tr>
</tbody>
</table>

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

\[
\langle A_1 \rangle \# \land \langle A_2 \rangle \# \land \text{"exactly one \#" is satisfiable} \iff L(A_1) \cap L(A_2) \neq \emptyset.
\]
Conclusion

Results:

- VLDL as expressive as $\omega$-VPA

<table>
<thead>
<tr>
<th></th>
<th>validity</th>
<th>model-checking</th>
<th>infinite games</th>
</tr>
</thead>
<tbody>
<tr>
<td>LTL</td>
<td>PSpace</td>
<td>PSpace</td>
<td>2ExpTime</td>
</tr>
<tr>
<td>LDL</td>
<td>PSpace</td>
<td>PSpace</td>
<td>2ExpTime</td>
</tr>
<tr>
<td>VLDL</td>
<td>ExpTime</td>
<td>ExpTime</td>
<td>3ExpTime</td>
</tr>
</tbody>
</table>

Using (deterministic) pushdown automata as guards leads to undecidability, i.e., $\langle A_1 \rangle \# \land \langle A_2 \rangle \# \land \text{"exactly one "} \# \text{ is satisfiable} \iff L(A_1) \cap L(A_2) \neq \emptyset$. 

Martin Zimmermann Saarland University Visibly Linear Dynamic Logic 17/17
Conclusion

Results:

- VLDL as expressive as $\omega$-VPA

<table>
<thead>
<tr>
<th></th>
<th>validity</th>
<th>model-checking</th>
<th>infinite games</th>
</tr>
</thead>
<tbody>
<tr>
<td>LTL</td>
<td>PSPACE</td>
<td>PSPACE</td>
<td>2ExpTime</td>
</tr>
<tr>
<td>LDL</td>
<td>PSPACE</td>
<td>PSPACE</td>
<td>2ExpTime</td>
</tr>
<tr>
<td>VLDL</td>
<td>ExpTime</td>
<td>ExpTime</td>
<td>3ExpTime</td>
</tr>
<tr>
<td>VLTL</td>
<td>ExpTime</td>
<td>ExpTime</td>
<td>?</td>
</tr>
</tbody>
</table>

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

\[
\langle A_1 \rangle \# \land \langle A_2 \rangle \# \land \text{“exactly one } \# \text{”}
\]

is satisfiable $\iff L(A_1) \cap L(A_2) \neq \emptyset$. 

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

\[
\langle A_1 \rangle \# \land \langle A_2 \rangle \# \land \text{“exactly one } \# \text{”}
\]

is satisfiable $\iff L(A_1) \cap L(A_2) \neq \emptyset$. 