Symbolic Model Checking
Property Specification Language*

Ji Wang
National Laboratory for Parallel and Distributed Processing
National University of Defense Technology

*Joint Work with
Wanwei Liu, Huowang Chen, Xiaodong Ma and Zhaofei Wang
Agenda

- Motivations
- Brief Overview of Symbolic Model Checking of LTL
- From PSL to APSL
- Tableau based Model Checking of APSL
- BDD based Encoding of APSL Tableaux
- Tool and Experiments
- Conclusion
Model checking is an automatic technique to verify if a system satisfies its design specification.

Why model checking?
- Complexity of modern hardware/software.
- Impractical for manually verification and proving.

How to run model checking?
- Model, abstraction of systems, described as (finite) transition systems, Kripke structures
- Specification, property to be verified
- Checking!
Specifications are written with various temporal logics.

**PSL (Property Specification Language)** has become an industrial standard (IEEE 1850).
- Evolved from some industrial used language, such as VHDL & Verilog.
- Has full expressiveness to describe all the omega-regular properties.

Properties such as “p holds at every even moment” cannot be expressed by any LTL formula.
Verification of PSL

- **Approaches**
  - Bustan, Fisman and Havlicek developed an automata-based approach for model checking PSL.
  - Tuerk, Schneider and Gordon presented PSL model checking using HOL and SMV.
  - Pnueli and Zaks developed a model checking approach, based on testers.

- **Tools**
  - The tool RuleBase of IBM
  - Zeroln of Mentor
Motivation

- How to efficiently model check PSL?

- **Symbolic Model Checking PSL**
  - A BDD-based symbolic approach for PSL model checking
  - Achieve the goal without doing too much adaptation to the existing popular verification tools
Basic Idea

- Property Specification Language.

- PSL = FL + OBE

- An extension of LTL

- CTL
Clarke et al presented an adapted LTL model checking framework, which converts the LTL MC problem to that of CTL.
A variant of PSL, namely APSL is presented, which has precisely the same expressiveness.

The tableau based symbolic model checking algorithm of LTL is extended to that of APSL.

Extend NuSMV tool, and make it support APSL.
Agenda

- Motivations
- Brief Overview of Symbolic Model Checking of LTL
- From PSL to APSL
- Tableau based Model Checking of APSL
- BDD based Encoding of APSL Tableaux
- Tool and Experiments
- Conclusion
A model is a state-labeled transition system (or, Kripke structure)

\[ M = \langle S, \rho, I, L, \Omega \rangle \]

where:
- \( S \) is a (finite) set of states.
- \( \rho \subseteq S \times S \), is the transition relation.
- \( I \subseteq S \), is a set of initial states.
- \( L: S \rightarrow 2^{AP} \), is the labeling function.
- \( \Omega \subseteq 2^S \), is a set of fairness constraints.
The linear perspective of a model

\[ \Omega = \{s_2, \{s_4\}\} \]
The branching perspective of a model

\[ \Omega = \{ \{s_2\}, \{s_4\} \} \]
Linear Temporal Logic

LTL

\[ \varphi :: = p \quad \text{proposition} \]

\[ \quad \neg \varphi \]

\[ \quad \varphi \land \varphi \]

\[ \quad X \varphi \quad \text{next} \]

\[ \quad \varphi U \varphi \quad \text{until} \]
\textbf{LTL Semantics}

- \( \pi, i \models p \) iff \( p \in \pi(i) \).
- \( \pi, i \models \neg \varphi \) iff \( \pi, i \not\models \varphi \) not holds.
- \( \pi, i \models \varphi_1 \land \varphi_2 \) iff \( \pi, i \models \varphi_1 \) and \( \pi, i \models \varphi_2 \).
- \( \pi, i \models X \varphi \) iff \( \pi, i + 1 \models \varphi \).
- \( \pi, i \models \varphi_1 U \varphi_2 \) iff there is some \( j \geq i \), s.t. \( \pi, j \models \varphi_2 \) and for each \( i \leq k < j \) s.t. \( \pi, k \models \varphi_1 \).

\textbf{Model checking problem of LTL}

- \( M \models \varphi \) iff all fair derived paths of \( M \) satisfy \( \varphi \).
CTL

Semantics of CTL formulae is defined on computation trees.

The CTL model checking problem $M \models \varphi$ is to verify that if all computation trees unwound from $M$ satisfy $\varphi$. 

$\varphi :: = p$

| $\neg \varphi$
| $\varphi \land \varphi$
| $AX \varphi$
| $A(\varphi \cup \varphi)$
| $E(\varphi \cup \varphi)$
CTL Model Checking

Framework of CTL model Checking (explicit)

- Model $M$
- CTL formula $\varphi$
- $\text{Sat set of } \varphi \text{ upon } M, [\varphi]_M$
- $I \subseteq [\varphi]_M$?
- Yes: Affirmative answer
- No: Counterexample
Idea of symbolic model checking

- Using $n$ bit variables to represent $2^n$ states.

We can use 2 bit variables $v_0$ and $v_1$ to encode the states.
Each subset of the state set corresponds to a Boolean function over $v_0$ and $v_1$. For example, the state set $\{s_1, s_2\}$, in which $p$ is evaluated to true, can be represented as $\neg v_1$. 
Each transition can be characterized as a Boolean over $v_0$, $v_1$, $v'_0$ and $v'_1$. E.g., the transition corresponding to the red edge can be written as $v_0 \land \neg v_1 \land v'_0 \land v'_1$. Use the disjunction of such formulae as the encoding of the transition relation.
- Initial states, fairness constraints and the labeling (for each proposition) can be encoded into a Boolean function.
- What we need to storage is a set of Boolean formulae, instead of explicit states and transitions.
- With these formulae, we can compute, in a bottom-up manner, the $\text{Sat}$ set of each subformula.
- This computing process is manipulated based on BDDs (Binary Decision Diagrams), which can be implemented in an efficient way.
Symbolic Model Checking of LTL

Model $M$

Transition system $M \parallel T_{\neg \varphi}$

$M \parallel T_{\neg \varphi} \vDash EG \text{ true?}$

LTL formula $\varphi$

Tableau $T_{\neg \varphi}$

Affirmative answer

Counterexample

Yes

No
Motivations

Brief Overview of Symbolic Model Checking of LTL

From PSL to APSL

Tableau based Model Checking of APSL

BDD based Encoding of APSL Tableaux

Tool and Experiments

Conclusion
PSL = FL + OBE

An extension of LTL

CTL
\( \varphi :: = b \)
\[\mid \neg \varphi \quad \mid \varphi \land \varphi \]
\[\mid X\varphi \quad \mid \varphi U \varphi \]
\[\mid \varphi \text{ abort } b \quad \text{ abort} \]
\[\mid r T \varphi \quad \text{ trigger} \]

Sequential Extended Regular Expressions (SERE)
$r ::= b$

| $r; r$ | concatenation |
| $r:r$ | fusion |
| $r || r$ | choice |
| $r && r$ | and |
| $r^*$ | Kleen closure |
| $r@c$ | clock sampling |
Examples

Concatenation: $r_1; r_2$

Fusion: $r_1: r_2$
Clock Sampling: $r@c$
Semantics of FL

- Formulae of $b$, $\neg \varphi$, $\varphi_1 \land \varphi_2$ are defined as usual
- Formulae of $X \varphi$ and $\varphi_1 U \varphi_2$ are defined as same as in LTL
- $\pi, i \models (\varphi \text{ abort } b)$ iff either $\pi, i \models \varphi$ or there is some $j \geq i$, and some $\pi'$, s.t. $\pi[i,j];\pi' \models \varphi$ and $\pi, j+1 \models b$
- $\pi, i \models rT \varphi$ iff there is some $j \geq i$, s.t. $\pi[i,j] \in L(r)$, and $\pi, j \models \varphi$
Symbolic model checking PSL?

- The major effort of PSL model checking must put on that for FL formulae.
- Explore the idea of LTL symbolic model checking.
For each formula $\varphi$, we can inductively construct the set of *elementary formulae* of $\varphi$, denoted $El(\varphi)$ as follows:

- $El(b) = \{p \in AP \mid p \text{ occurs in } b\}$;
- $El(\neg \psi) = El(\psi)$;
- $El(\psi_1 \land \psi_2) = El(\psi_1) \cup El(\psi_2)$;
- $El(X\psi) = \{X\psi\} \cup El(\psi)$;
- $El(\psi_1 U \psi_2) = \{X(\psi_1 U \psi_2)\} \cup El(\psi_1) \cup El(\psi_2)$

An element of $El(\varphi)$ is either an atomic proposition or a formula of the form $X\psi$
For each subformula $\psi$ of $\varphi$, define the function $Sat$, which maps $\psi$ to a set of subsets of $El(\varphi)$.

- $Sat(p) = \{ W \subseteq El(\varphi) \mid p \in W \}$;
- $Sat(\neg \psi) = 2^{El(\varphi)} \setminus Sat(\psi)$;
- $Sat(\psi_1 \land \psi_2) = Sat(\psi_1) \cap Sat(\psi_2)$;
- $Sat(\Box \psi) = \{ W \subseteq El(\varphi) \mid \Box \psi \in W \}$;
- $Sat(\psi_1 U \psi_2) = Sat(\psi_2) \cup (Sat(\psi_1) \cap Sat(\Box(\psi_1 U \psi_2)))$. 
The tableau of $\varphi$, denoted $T_\varphi$ is the transition system $\langle S_\varphi, \rho_\varphi, I_\varphi, L_\varphi, \Omega_\varphi \rangle$, where:

- $S_\varphi$ consists of subsets of $El(\varphi)$.
- $(W, W') \in \rho_\varphi$ iff for each $X \psi \in El(\varphi)$, $W \in Sat(X \psi)$ if and only if $W' \in Sat(\psi)$.
- $I_\varphi = Sat(\varphi)$.
- $L_\varphi(W) = W \cap AP$.
- For each subformula $\psi_1 U \psi_2$ of $\varphi$, there is a fairness constraint $Sat(\neg (\psi_1 U \psi_2)) \cup Sat(\psi_2)$ in $\Omega_\varphi$.

For each $\pi \in (2^{AP})^\omega$, $\pi \models \varphi$ iff $\pi \in L(T_\varphi)$.
For FL formula, the difficulty is that the transition structure is not explicit.
So, when defining the Sat function for formula of \( r T b \), it is hard to write an explicit formula.

Replace SEREs with NFAs
A variant of PSL, namely APSL.

APSL = AFL + OBE

A variant of FL

CTL
AFL Syntax

\[ \varphi :: = b \]
\[ \quad | \quad \neg \varphi \quad | \quad \varphi \wedge \varphi \]
\[ \quad | \quad X\varphi \quad | \quad \varphi U \varphi \]
\[ \quad | \quad A \text{ abort! } b \quad \text{ strongly abort} \]
\[ \quad | \quad A T \varphi \quad \text{automaton trigger} \]
The semantics

- $\pi, i \models A \text{ abort! } b$ iff there is some $w \in PreL(A)$ and some $j \geq i$, s.t. $\pi[i,j] = w$ and $\pi, j+1 \models b$.
- $\pi, i \models AT \varphi$ iff there is some $j \geq i$, s.t. $\pi[i,j] \in L(A)$ and $\pi, j \models \varphi$.

- AFL and FL have precisely the same expressiveness.
- Study symbolic model checking problem for AFL.
Agenda

- Motivations
- Brief Overview of Symbolic Model Checking of LTL
- From PSL to APSL
- Tableau based Model Checking of APSL
- BDD based Encoding of APSL Tableaux
- Tool and Experiments
- Conclusion
Given an NFA $A = \langle \Sigma, Q, \delta, Q_0, F \rangle$, we denote by $A^q$ the NFA $\langle \Sigma, Q, \delta, \{q\}, F \rangle$.

- Elementary formula set of AFL formula
  - For the formulae of $b$, $\neg \psi$, $\psi_1 \land \psi_2$, $X \psi$, $\psi_1 \cup \psi_2$, their elementary formula sets are defined same as before.
  - $El(A \text{ abort! } b) = El(b) \cup \{X(A^q \text{ abort! } b) \mid q \text{ is a state of } A\}$.
  - $El(A \text{ T } \psi) = El(\psi) \cup \{X(A^q \text{ T } \psi) \mid q \text{ is a state of } A\}$. 
- Sat function of abort!

\( \text{Sat}(A^q \text{ abort! } b) \)

\[ = \]

\[ \text{Sat}(b) \cup \{ \mathcal{W} \subseteq \text{El}(\varphi) \mid \exists r \in \delta(q, \mathcal{W} \cap \text{AP}), \text{s.t. } X(A^r \text{ abort! } b) \in \mathcal{W} \} \]

\[ \pi, i \models A^q \text{ abort! } b \]

iff

either \( \pi, i \models b \), or \( \pi, i \models X(A^r \text{ abort! } b) \) for some \( r \in \delta(q, \pi(i)) \)
 Sat function of \( T \)

\[
\text{Sat}(A^q T \psi) = \\
\{ W \in \text{Sat}(\psi) \mid \delta (q, W \cap \text{AP}) \cap F \neq \emptyset \} \cup \\
\{ W \subseteq \text{El}(\varphi) \mid \exists r \in \delta (q, W \cap \text{AP}), \text{ s.t. } X(A^r T \psi) \in W \}
\]

\( \pi, i \models A^q T \psi \) iff

either \( \pi, i \models \psi \) and \( \delta (q, \pi(i)) \cap F \neq \emptyset \)

or \( \pi, i \models X(A^r T \psi) \) for some \( r \in \delta (q, \pi(i)) \)
Given an AFL formula $\varphi$, suppose that $(A_1, \psi_1)$, ..., $(A_m, \psi_m)$ are all its trigger pairs, and $A_i$'s state set is $Q_i$. The tableau of $\varphi$ is a special transition system

$$T_{\varphi} = \langle S_{\varphi}, \rho_{\varphi}, I_{\varphi}, L_{\varphi}, \Omega_{\varphi} \rangle$$

- $S_{\varphi}$ consists of tuples $\langle W, (P_1, \ldots, P_m) \rangle$, where $W \subseteq El(\varphi)$ and $P_j \subseteq Q_j$.
- $\rho_{\varphi}$
  - $(\langle W, (P_1, \ldots, P_m) \rangle, \langle W', (P'_1, \ldots, P'_m) \rangle) \in \rho_{\varphi}$ if
    - For each $X\psi \in El(\varphi)$, $W \in \text{Sat}(X\psi)$ iff $W' \in \text{Sat}(\psi)$;
    - For each $1 \leq j \leq m$, $((W, P_j), (W', P'_j)) \in \rho_{(A_j, \psi_j)}$.
- $I_{\varphi} = \{ \langle W, (P_1, \ldots, P_m) \rangle \mid W \in \text{Sat}(\varphi) \}$.
- $L_{\varphi}(\langle W, (P_1, \ldots, P_m) \rangle) = W \cap AP$. 
Given an AFL formula \( \varphi \), we say that \((A, \psi)\) is a **trigger pair** in \( \varphi \), if there is some \( q \), such that \( A^q T \psi \) is a subformula of \( \varphi \).

Assuming \( A = \langle 2^{AP}, Q, \delta, Q_0, F \rangle \), the trigger pair \((A, \psi)\) of \( \varphi \) derives a **trigger transition relation** \( \rho_{(A, \psi)} \subseteq (2^{El(\varphi)} \times 2^Q)^2 \) such that \((W_1, Q_1, W_2, Q_2) \in \rho_{(A, \psi)}\) iff

- If \( Q_1 = \emptyset \), then \( Q_2 = \{ q \mid W_2 \in \text{Sat}(A^q T \psi) \} \).
- If \( Q_1 \neq \emptyset \), then for each \( q \in Q_1 \),
  - either \( W_1 \in \text{Sat}(\psi) \) and \( \delta(q, W_1 \cap AP) \cap F \neq \emptyset \)
  - or there is some \( q' \in Q_2 \), s.t. \( q' \in \delta(q, W_1 \cap AP) \)
For each trigger pair \((A, \psi)\) and each state \(q\) of \(A\), and each \(s \in \text{Sat} (A^q T \psi)\), a fair path starting from \(s\) must satisfy \(A^q T \psi\).
\( \Omega_\varphi \) consists of three parts:

- For each subformula \( \psi_1 \cup \psi_2 \), create a fairness constraint
  \[
  \{ \langle W, (P_1, \ldots, P_m) \rangle \mid W \in \text{Sat}(\psi_2) \text{ or } W \notin \text{Sat}(\psi_1 \cup \psi_2) \}.
  \]

- For each subformula \( A^q \text{ abort! } b \), create a fairness constraint
  \[
  \{ \langle W, (P_1, \ldots, P_m) \rangle \mid \text{W} \in \text{Sat}(b) \text{ or } W \notin \bigcup_{r \in Q} \text{Sat}(A^r \text{ abort! } b) \}.
  \]

- For each trigger pair \( (A_i, \psi_i) \), create a fairness constraint
  \[
  \{ \langle W, (P_1, \ldots, P_m) \rangle \mid P_i = \emptyset \}.
  \]
Theorem (Language Property of AFL tableaux):

For each $\pi \in (2^{AP})^\omega$, $\pi \models \varphi$ iff $\pi \in L(T_\varphi)$

AFL model checking problem is converted to that of CTL.

$M \models \varphi$ iff $(M\parallel T_\varphi)$ does not satisfy EG true
Agenda

- Motivations
- Brief Overview of Symbolic Model Checking of LTL
- From PSL to APSL
- Tableau based Model Checking of APSL
- BDD based Encoding of APSL Tableaux
- Tool and Experiments
- Conclusion
With the BDD based technique, product of two transition systems can be naturally implemented.

BDD encoding of the original model can be acquired from the users’ input.

How to obtain the encoding of an AFL formula’s tableau.
Given an AFL formula $\varphi$ with trigger pairs $(A_1, \psi_1), \ldots, (A_m, \psi_m)$ and the state set of $A_i$ is $Q_i$.

Recall that a state in the tableau is a tuple $\langle W, (P_1, \ldots, P_m) \rangle$, where $W \subseteq El(\varphi)$ and $P_j \subseteq Q_j$. Then:

- For each $\psi \in El(\varphi)$, create a Boolean variable $u_\psi$.
- For each $1 \leq i \leq m$ and each $q \in Q_i$, create a Boolean variable $v_{(i, q)}$.

For each subformula $\psi$ of $\varphi$, we may build a Boolean formula $f_\psi$, which characterizes $\text{Sat}(\psi)$. 
The symbolic encoding of transition relation is the conjunction of the following issues:

- For each $X\psi \in El(\varphi)$, add a conjunct $u_{X\psi} \Leftrightarrow f_\psi$.
- For each $1 \leq i \leq m$, employ the following two conjuncts

$$
\left( \bigwedge_{q \in Q_i} \neg v(i,q) \right) \Rightarrow \bigwedge_{q \in Q_i} \left( v'(i,q) \Leftrightarrow f'_{A_i^q T\psi_i} \right)
$$

If $Q_1 = \emptyset$, then $Q_2 = \{ q \mid W_2 \in \text{Sat}(A^q T\psi) \}$
For each subformula $\psi_1 \mathsf{U} \psi_2$, we add a fairness constraint encoding

$$\neg f_{\psi_1 \mathsf{U} \psi_2} \lor f_{\psi_2}.$$ 

For each subformula $A^q \textbf{abort}! b$ with $A = \langle \Sigma, Q, \delta, Q_0, F \rangle$, we add the fairness constraint encoding

$$f_b \lor \bigwedge_{q' \in Q} \neg f_{A^q \textbf{abort}! b}.$$ 

For each trigger pair $(A_i, \psi_i)$, we create the fairness constraint encoding

$$\bigwedge_{q \in Q_i} \neg \nu(i,q).$$
Agenda

- Motivations
- Brief Overview of Symbolic Model Checking of LTL
- From PSL to APSL
- Tableau based Model Checking of APSL
- BDD based Encoding of APSL Tableaux
- Tool and Experiments
- Conclusion
Tool Support

- From NuSMV to ENuSMV
  
  NuSMV is a symbolic model checking tool by CMU/ict-IRST, and it supports both CTL and LTL model checking.

  - The extended version of SMV is adapted from NuSMV Ver 2.4.3.
  - ENuSMV Ver 1.0 support ETL model checking, and Ver 1.1 support APSL.
  - Available at [http:// enusmv.sourceforge.net](http:// enusmv.sourceforge.net)
Defining Automata constructs in ENuSMV

CONNECTIVE $A(a_1,a_2)$

STATES:
\[ > q_1, q_2 < \]

TRANSITIONS($q_1$)

\[ \text{case} \]
\[ a_1: \{q_1,q_2\}; \]
\[ a_2: q_2; \]
\[ \text{esac;} \]
Philosopher Dining

- **Feasibility**: each philosopher can possibly have a meal.
- **Liveness**: it is possible for a philosopher to eat infinitely many times.

**Machine Specification**
- CPU: Intel Core Duo2 (2.66GHz)
- Memory-size: 2G
## Feasibility

<table>
<thead>
<tr>
<th>#philosophers</th>
<th># states</th>
<th>#time (sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>CTL</td>
</tr>
<tr>
<td>6</td>
<td>1135</td>
<td>1135</td>
</tr>
<tr>
<td>7</td>
<td>3545</td>
<td>3545</td>
</tr>
<tr>
<td>8</td>
<td>11395</td>
<td>11395</td>
</tr>
<tr>
<td>#philosopher</td>
<td># states</td>
<td>#time (sec.)</td>
</tr>
<tr>
<td>--------------</td>
<td>----------</td>
<td>--------------</td>
</tr>
<tr>
<td></td>
<td>CTL</td>
<td>AFL</td>
</tr>
<tr>
<td>6</td>
<td>1135</td>
<td>4412</td>
</tr>
<tr>
<td>7</td>
<td>3545</td>
<td>14180</td>
</tr>
<tr>
<td>8</td>
<td>11395</td>
<td>45580</td>
</tr>
</tbody>
</table>
Consider the model of “accumulative carry circuit”, which is a synchronous circuit composed of a set of modulo 2 counters. These counters are connected in a series manner — the first counter’s input is set to 1, and the \((i+1)\)-th counter’s input is connected to the \(i\)-th counter’s output.
MOUDLE counter(carry_in)

VAR
  value: boolean;
  pre_value: boolean;

ASSIGN
  init(value):=0;
  init(pre_value) :=0;
  next(pre_value) :=value;
  next(value) := (value +carry_in) mod 2;

DEFINE
  carry_out := (pre_value & carry_in);
Accumulative Carry Circuit

bit_0: counter(1);
...
bit_i+1: counter(bit_i.output);
...
bit_n: counter(bit_n-1.output);
Consider the following properties:

- **Periodicity**: bit_0 carries out at each odd moment.
- **Monitoring**: once the waveform matches the pattern \( \text{true};(\text{bit}_0.\text{carry}_\text{out})^* \), then the signal bit_1.\text{carry}_\text{out} would raise in the next two steps.

None of the above properties is star-free.

Cannot be verified with the previous NuSMV.
## Accumulative Carry Circuit

<table>
<thead>
<tr>
<th>#bits</th>
<th># states</th>
<th>#time</th>
<th>#bits</th>
<th># states</th>
<th>#time</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>66560</td>
<td>0.011</td>
<td>7</td>
<td>8320</td>
<td>0.009</td>
</tr>
<tr>
<td>8</td>
<td>132096</td>
<td>0.012</td>
<td>8</td>
<td>16512</td>
<td>0.011</td>
</tr>
<tr>
<td>9</td>
<td>263168</td>
<td>0.019</td>
<td>9</td>
<td>32896</td>
<td>0.018</td>
</tr>
<tr>
<td>10</td>
<td>525312</td>
<td>0.024</td>
<td>10</td>
<td>65664</td>
<td>0.021</td>
</tr>
</tbody>
</table>
To test the scalability, we have done DME (Distributed Mutual Exclusive) circuit experiment. Results show that ENuSMV can handle models with more than $5 \times 10^{21}$ reachable states.

The experiment on security policy of SE-Linux shows that ENuSMV can handle a model larger than 120,000 lines.

Notably, for the same property, AFL model checking sometimes has a better performance than that of LTL, especially when the connective nesting depth is large.
Agenda

- Motivations
- Brief Overview of Symbolic Model Checking of LTL
- From PSL to APSL
- Tableau based Model Checking of APSL
- BDD based Encoding of APSL Tableaux
- Tool and Experiments
- Conclusion
Conclusion

- We have presented a variant of PSL, namely APSL, which has precisely the same expressive power as omega-regular expression.
- We extended the BDD-based symbolic model checking algorithm to that of APSL.
- A symbolic model checker supporting APSL is designed and implemented based on NuSMV.
Thank you.

Questions and Comments?