# 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

- 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!

# Specification Language

- 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
  - ZeroIn of Mentor

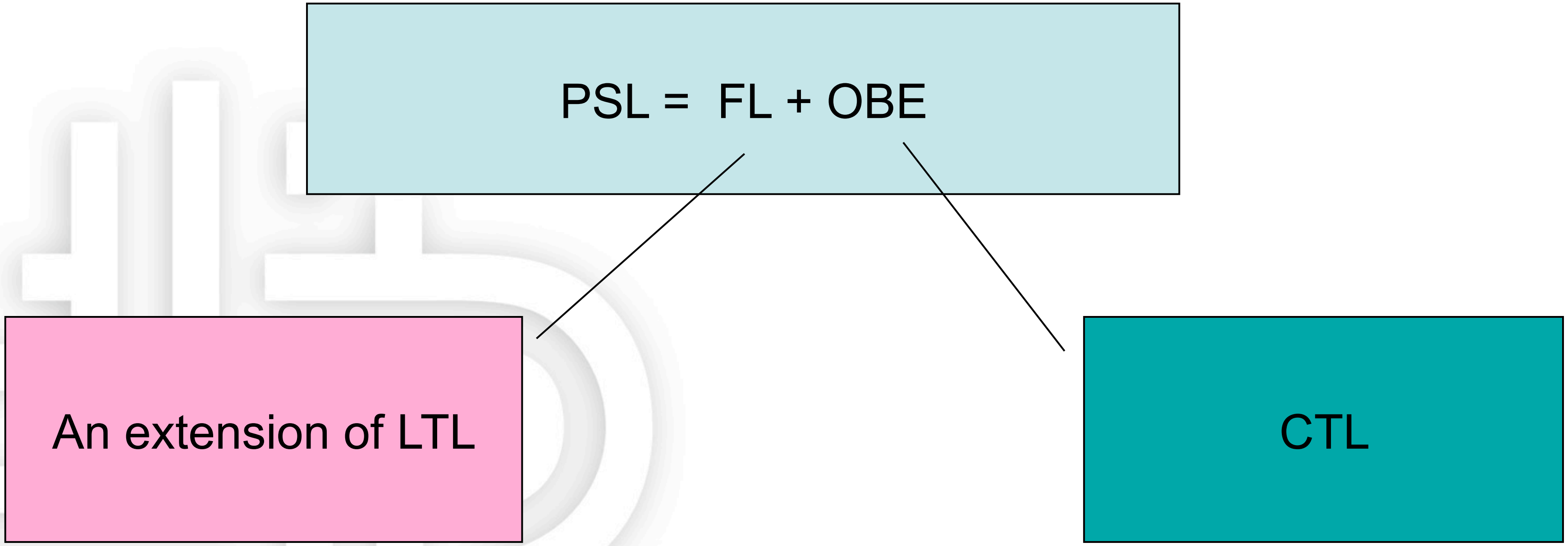
# Motivation

- 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

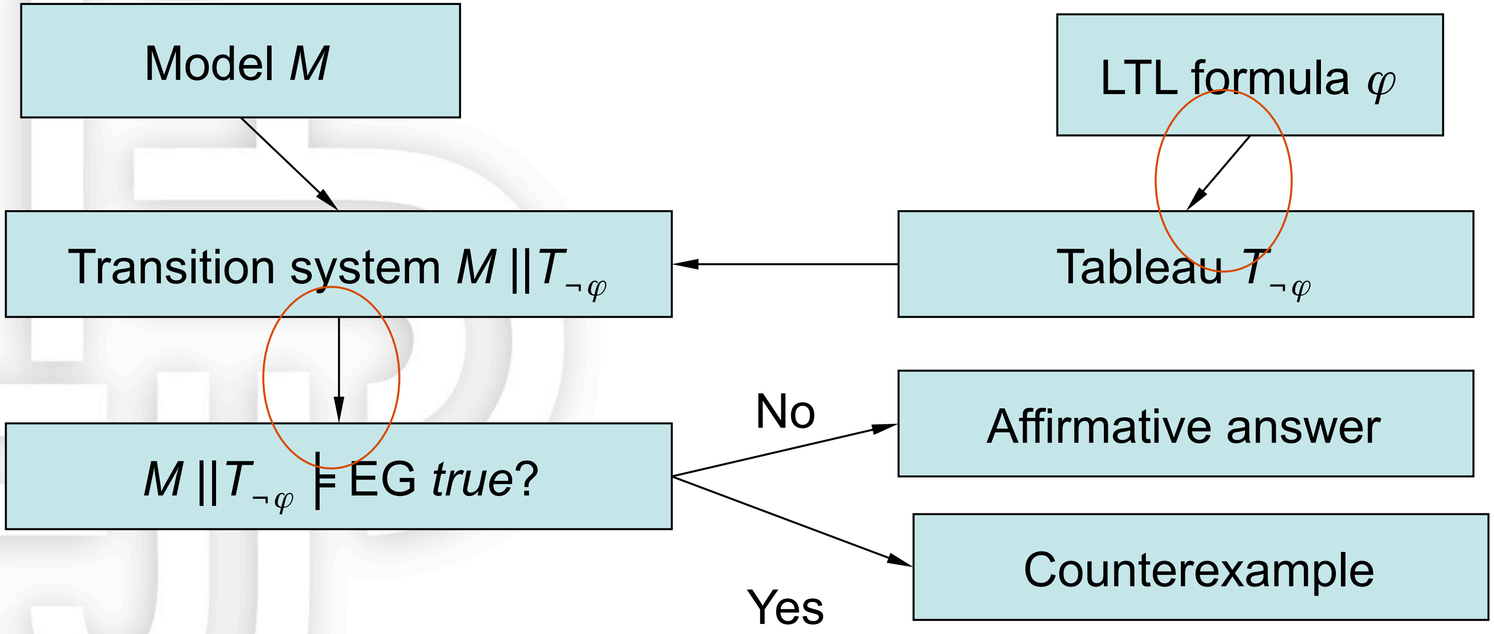
# Basic Idea

❖ Property Specification Language.



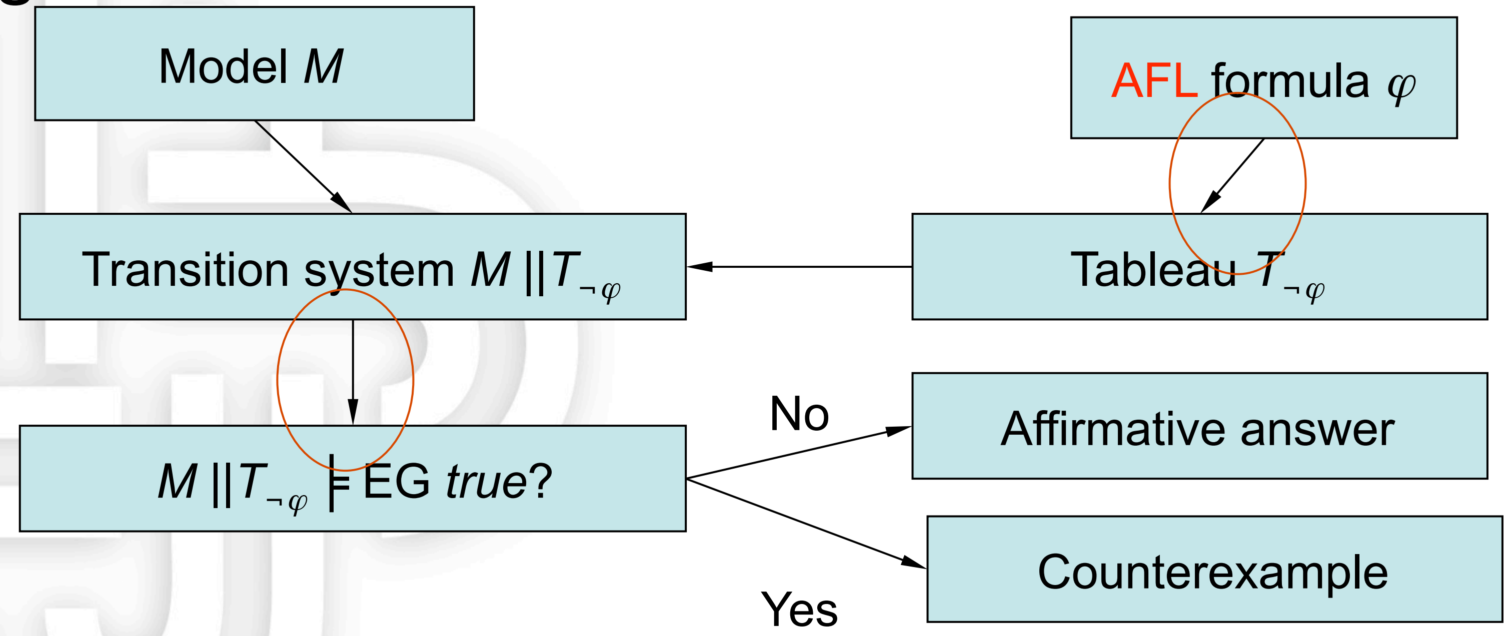
# Basic Idea

- Clarke et al presented an adapted LTL model checking framework, which converts the LTL MC problem to that of CTL



# Contributions

- 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

# Models

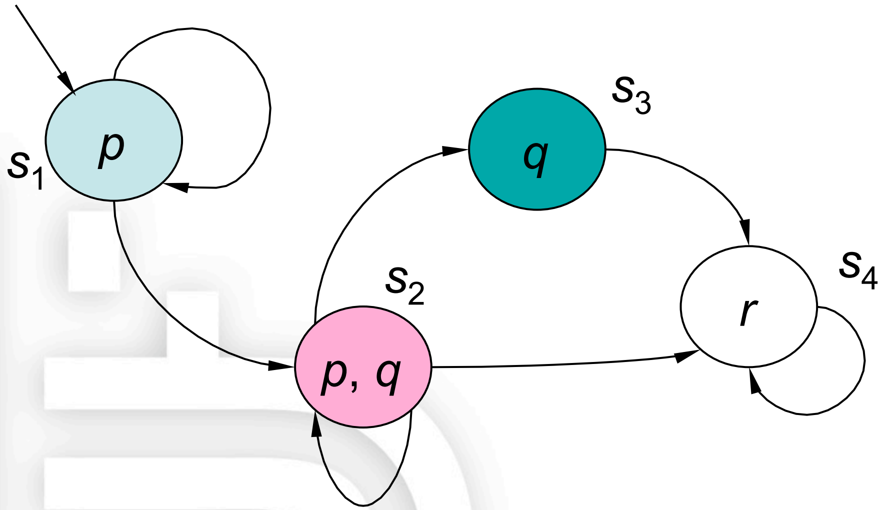
- A model is a state-labeled transition system (or, Kripke structure)

$$M = \langle S, \rho, I, L, \Omega \rangle$$

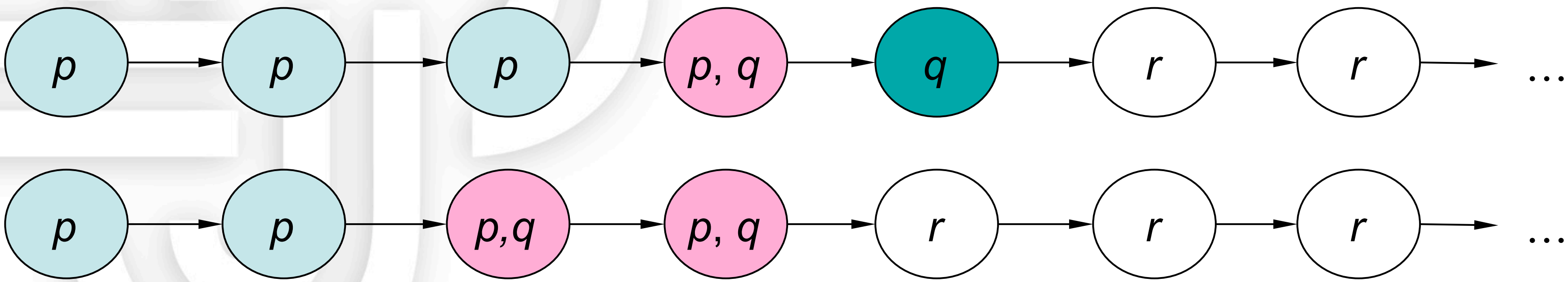
where:

- $S$ 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

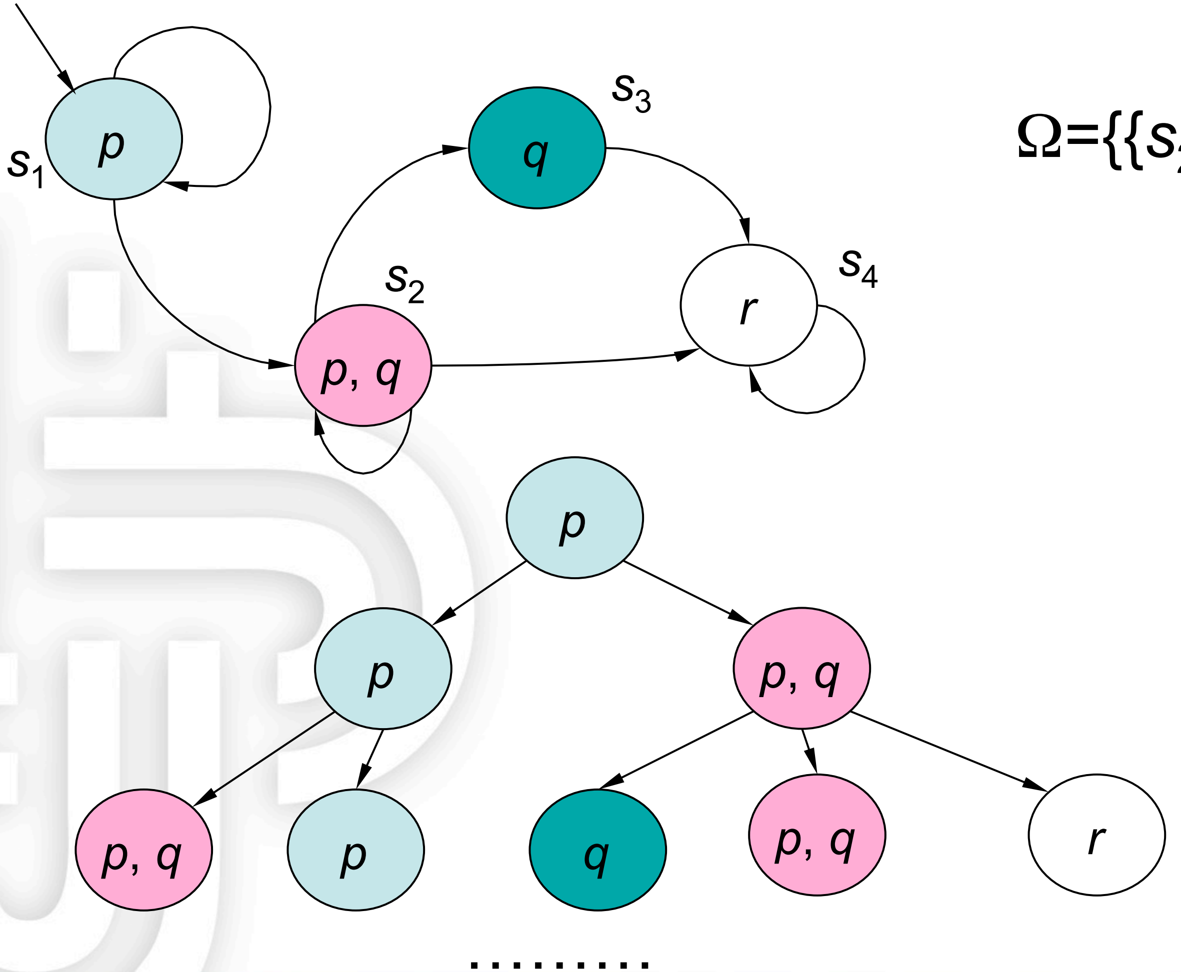



$\Omega=\{\{s_2\}, \{s_4\}\}$



# Computations

The branching perspective of a model




$\Omega=\{\{s_2\}, \{s_4\}\}$

# Linear Temporal Logic

❖ LTL

$$
\begin{array}{llll}
\varphi ::= p & & & \text{proposition} \\
 & \mid & \neg \varphi & \\
 & \mid & \varphi \wedge \varphi & \\
 & \mid & \mathsf{X}\varphi & \text{next} \\
 & \mid & \varphi \,\mathsf{U}\, \varphi & \text{until}
\end{array}
$$

# LTL Semantics

- LTL Semantics
  - $\pi, i \models p$ iff $p \in \pi(i)$.
  - $\pi, i \models \neg\varphi$ iff $\pi, i \models \varphi$ not holds.
  - $\pi, i \models \varphi_1 \wedge \varphi_2$ iff $\pi, i \models \varphi_1$ and $\pi, i \models \varphi_2$.
  - $\pi, i \models \mathrm{X}\varphi$ iff $\pi, i+1 \models \varphi$.
  - $\pi, i \models \varphi_1 \mathrm{U} \varphi_2$ iff there is some $j \geq i$, s.t. $\pi, j \models \varphi_2$ and for each $i \leq k < j$ s.t. $\pi, k \models \varphi_1$.
- Model checking problem of LTL
  - $M \models \varphi$ iff all fair derived paths of $M$ satisfy $\varphi$.

# Computation Tree Logic

❖ CTL

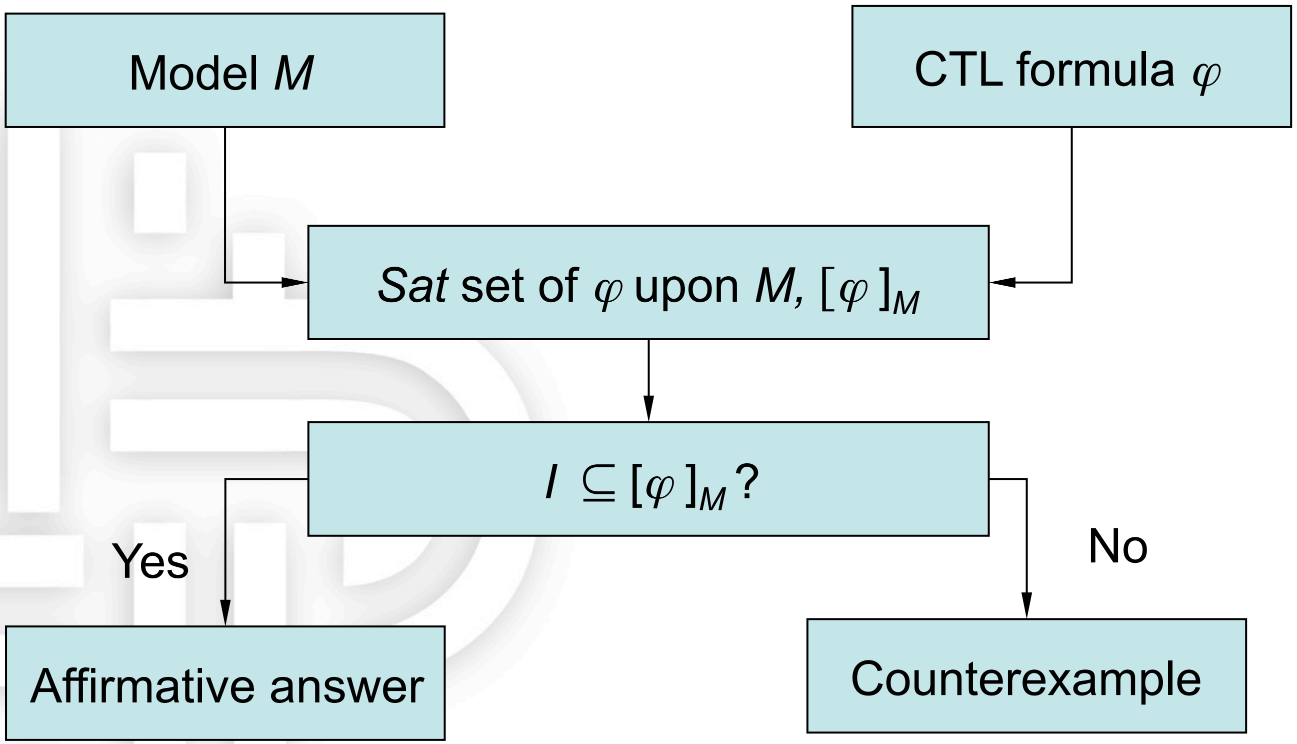
$$
\begin{aligned}
\varphi \; ::= \; & p \\
| \; & \neg\varphi \\
| \; & \varphi \wedge \varphi \\
| \; & \mathrm{AX}\varphi \\
| \; & \mathrm{A}(\varphi \,\mathrm{U}\, \varphi) \\
| \; & \mathrm{E}(\varphi \,\mathrm{U}\, \varphi)
\end{aligned}
$$

Semantics of CTL formulae is defined on computation trees.

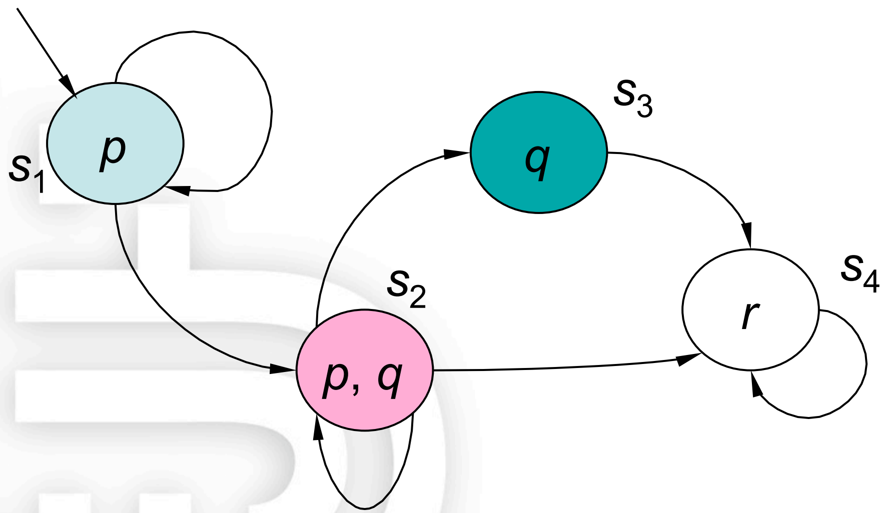
The CTL model checking problem $M \models \varphi$ is to verify that if all computation trees unwounded from $M$ satisfy $\varphi$.

# CTL Model Checking

- Framework of CTL model Checking (explicit)

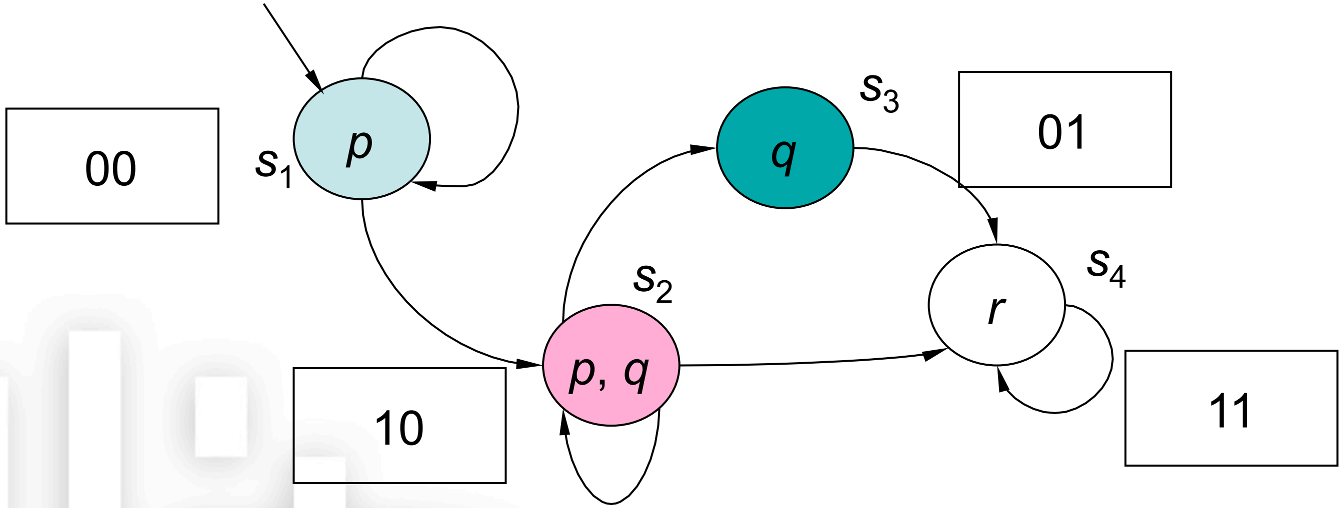


- Idea of symbolic model checking
  - Using $n$ bit variables to represent $2^n$ states.



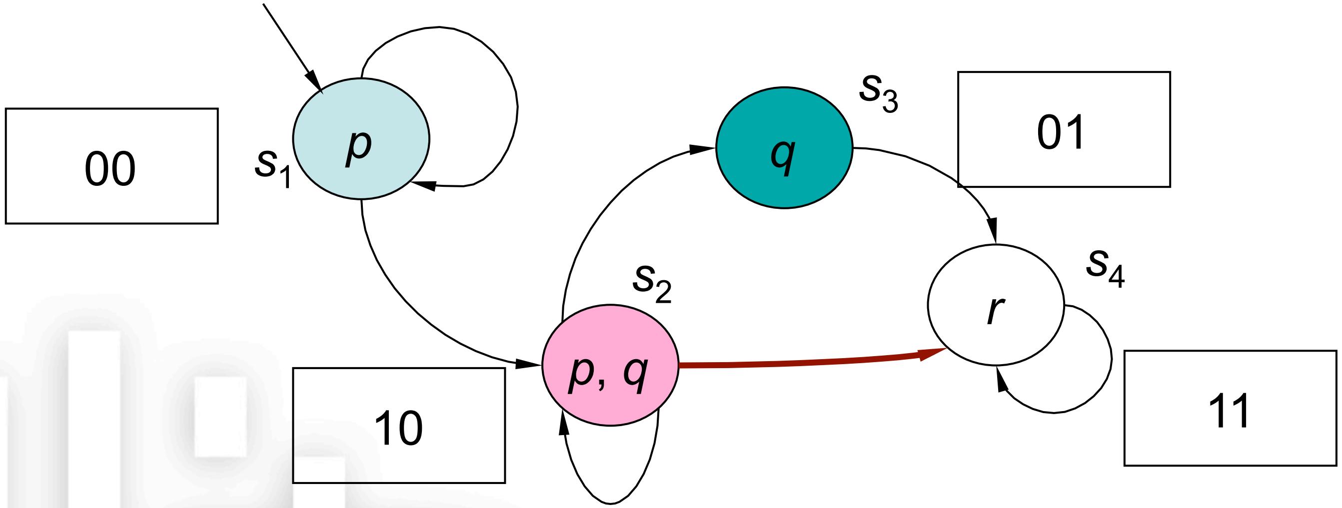

We can use 2 bit variables $v_0$ and $v_1$ to encode the states.

# CTL Symbolic Model Checking




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 $\neg v_1$.

# CTL Symbolic Model Checking

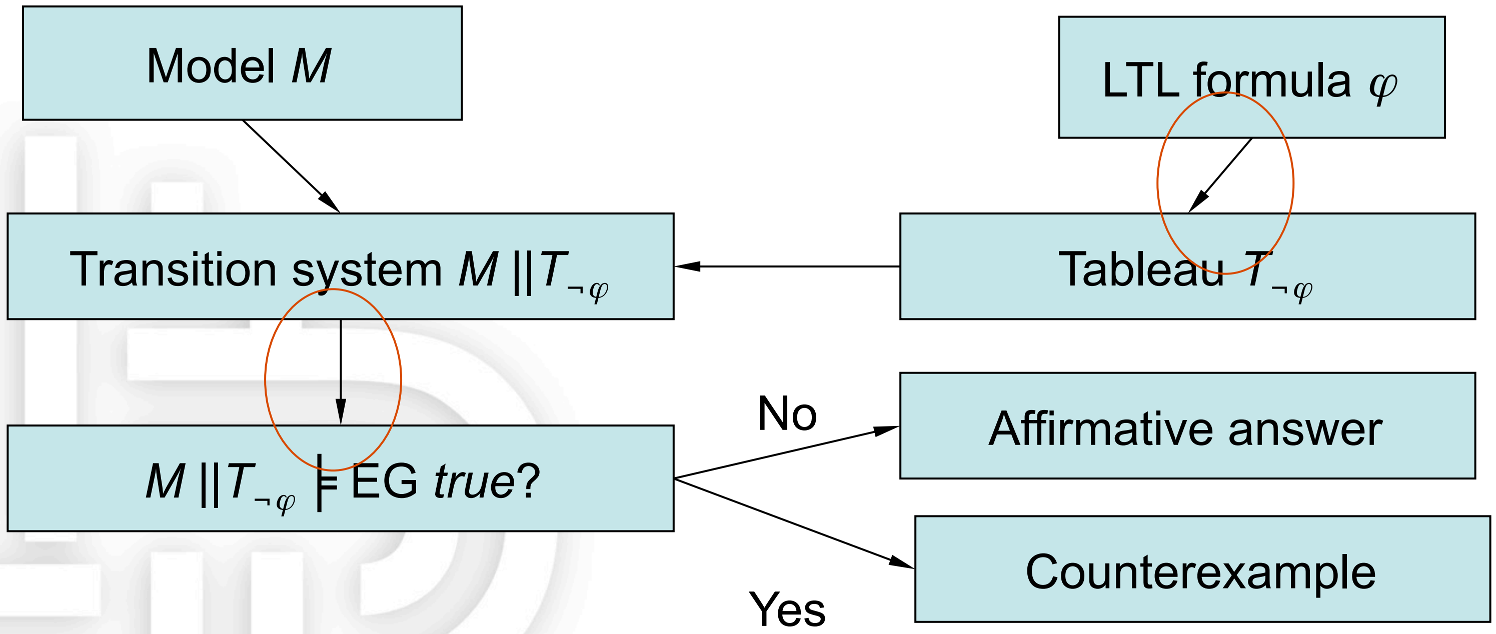



Each transition can be characterized as a Boolean over $v_0$, $v_1$, $v'_0$ and $v'_1$. e.g., the transition corresponding to the red edge can be written as $v_0 \land \neg v_1 \land v'_0 \land v'_1$.
Use the disjunction of such formulae as the encoding of the transition relation.

# CTL Symbolic Model Checking

- 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 *Sat* set of each subformula.
- This computing process is manipulated based on BDDs (Binary Decision Diagrams), which can be implemented in an efficient way.

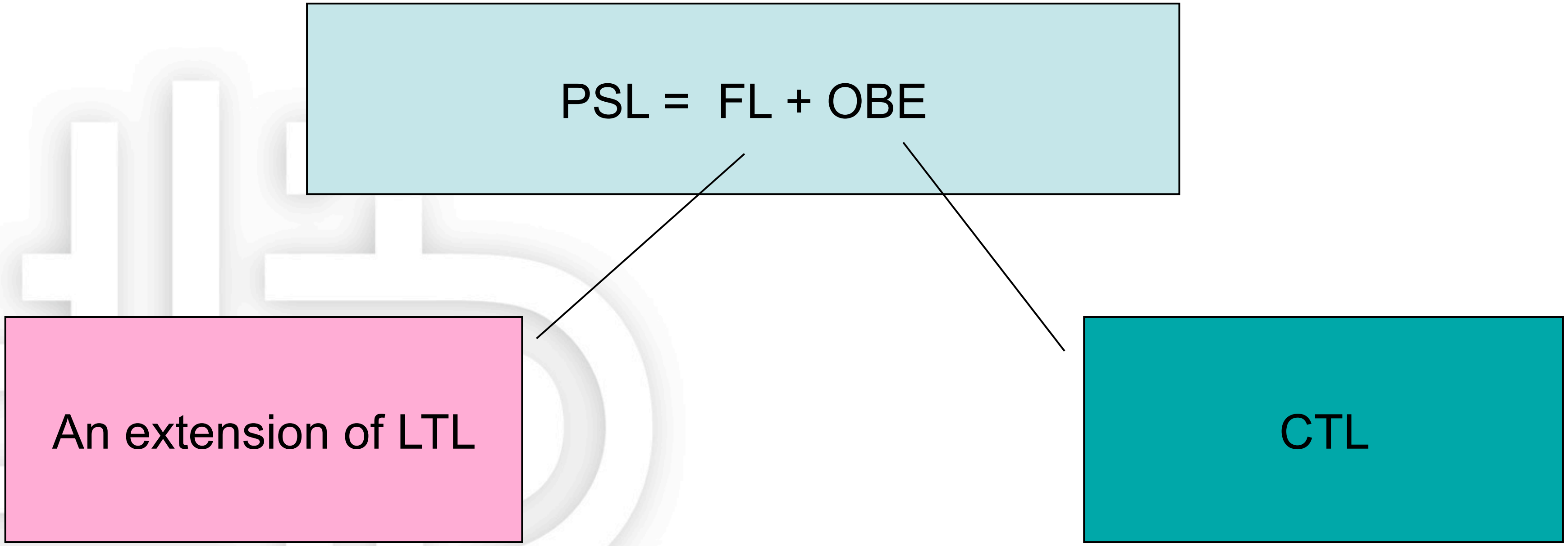
# Symbolic Model Checking of LTL



# 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

# PSL



# FL Syntax

$\varphi ::= b$

$\quad | \; \neg\varphi \qquad | \; \varphi \wedge \varphi$

$\quad | \; X\varphi \qquad | \; \varphi \, U \varphi$

$\quad | \; \varphi$ abort $b$ — abort

$\quad | \; r \, T \, \varphi$ — trigger

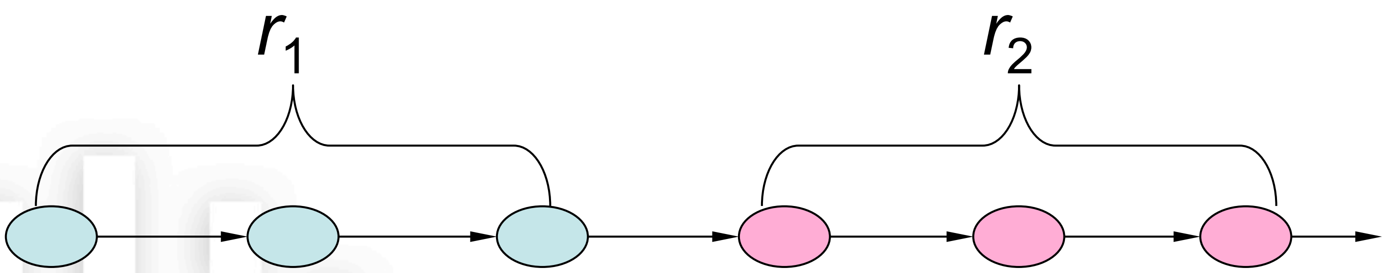
Sequential Extended Regular Expressions (SERE)

# SERE

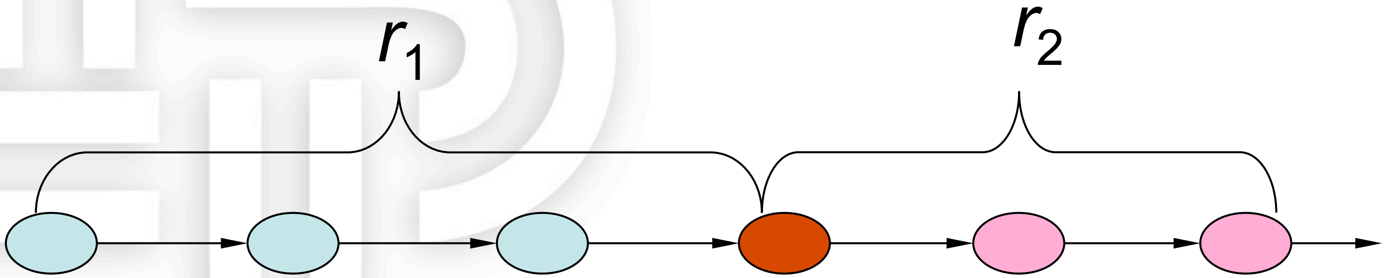
| | |
|---|---|
| $r ::= b$ | Boolean expression |
| $\mid r; r$ | *concatenation* |
| $\mid r{:}r$ | fusion |
| $\mid r \,\|\|\, r$ | choice |
| $\mid r \,\&\&\, r$ | and |
| $\mid r^{*}$ | Kleen closure |
| $\mid r@c$ | clock sampling |

# Examples

Concatenation: $r_1; r_2$

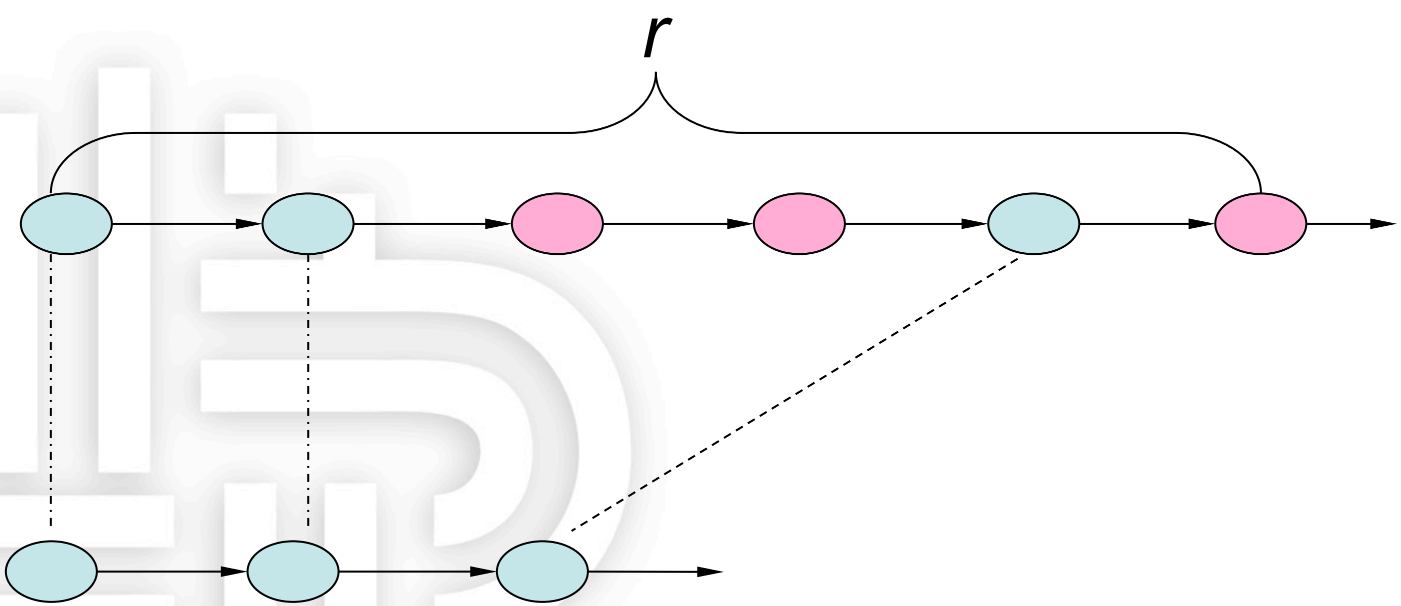



Fusion: $r_1: r_2$



# Examples

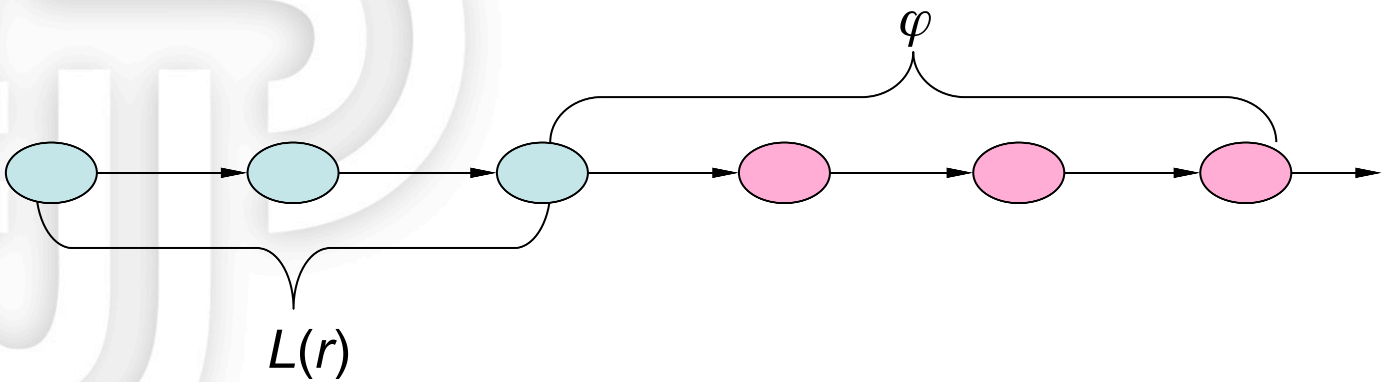
Clock Sampling: *r@c*



# Semantics of FL

- Semantics of FL formulae
  - Formulae of $b$, $\neg\varphi$, $\varphi_1 \wedge \varphi_2$ are defined as usual
  - Formulae of $\mathrm{X}\varphi$ and $\varphi_1 \mathrm{U} \varphi_2$ are defined as same as in LTL
  - $\pi, i \models (\varphi$ 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 $\pi, j+1 \models b$
  - $\pi, i \models r\ \mathrm{T}\,\varphi$ iff there is some $j \geq i$, s.t. $\pi[i,j] \in L(r)$, and $\pi, j \models \varphi$

$r\ \mathrm{T}\,\varphi$



# Symbolic model checking PSL?

- 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.

# Elementary Formulas

- For each formula $\varphi$, we can inductively construct the set of elementary formulae of $\varphi$, denoted $El(\varphi)$ as follows:
  - $El(b) = \{p \in AP \mid p \text{ occurs in } b\}$;
  - $El(\neg\psi) = El(\psi)$;
  - $El(\psi_1 \wedge \psi_2) = El(\psi_1) \cup El(\psi_2)$;
  - $El(\mathsf{X}\psi) = \{\mathsf{X}\psi\} \cup El(\psi)$;
  - $El(\psi_1 \mathsf{U} \psi_2) = \{\mathsf{X}(\psi_1 \mathsf{U} \psi_2)\} \cup El(\psi_1) \cup El(\psi_2)$
- An element of $El(\varphi)$ is either an atomic proposition or a formula of the form $\mathsf{X}\psi$

# *Sat* functions

- For each subformula $\psi$ of $\varphi$, define the function $Sat$, which maps $\psi$ to a set of subsets of $El(\varphi)$.

  – $Sat(p) = \{W \subseteq El(\varphi) \mid p \in W\}$;
  
  – $Sat(\neg\psi) = 2^{El(\varphi)} \backslash Sat(\psi)$;
  
  – $Sat(\psi_1 \wedge \psi_2) = Sat(\psi_1) \cap Sat(\psi_2)$;
  
  – $Sat(\mathrm{X}\psi) = \{W \subseteq El(\varphi) \mid \mathrm{X}\psi \in W\}$;
  
  – $Sat(\psi_1 \mathrm{U} \psi_2) =$
  $Sat(\psi_2) \cup (Sat(\psi_1) \cap Sat(\mathrm{X}(\psi_1 \mathrm{U} \psi_2)))$.

# Tableau

- 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 $\pi \in L(T_\varphi)$

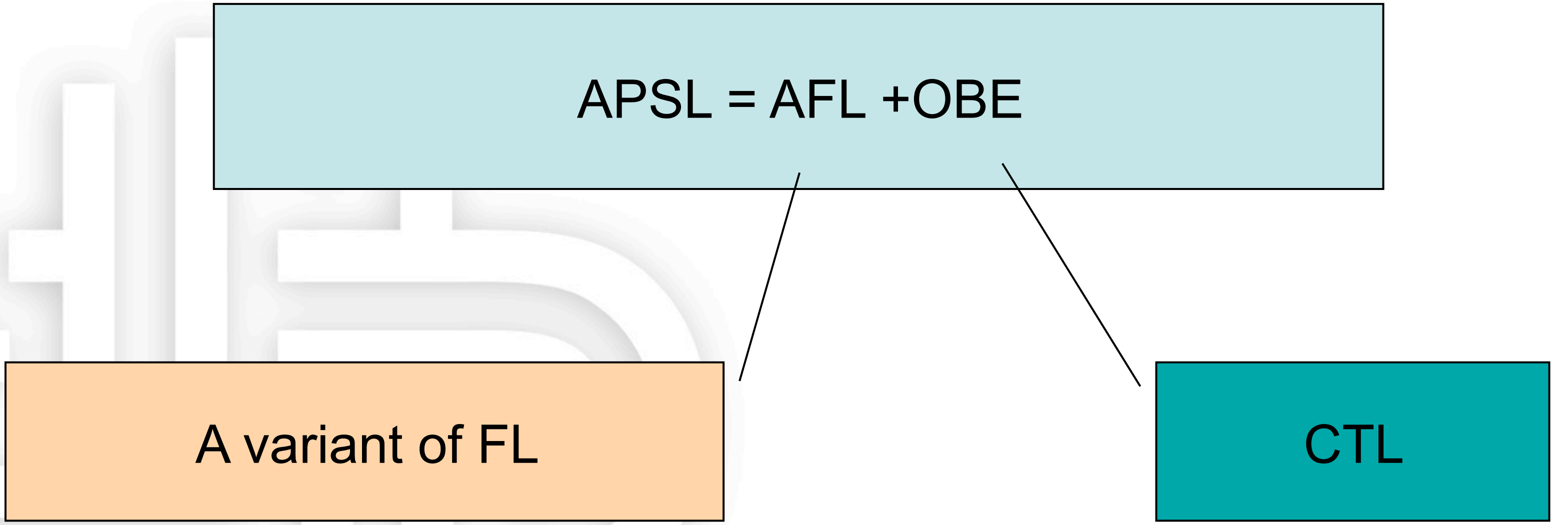
# Tableau of FL?

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

# APSL

- A variant of PSL, namely APSL.



# AFL Syntax

$$
\begin{aligned}
\varphi ::= {} & b \\
& \mid \neg\varphi \mid \varphi \wedge \varphi \\
& \mid X\varphi \quad \mid \varphi\, U\, \varphi \\
& \mid A\ \text{abort!}\ b && \text{strongly abort} \\
& \mid A\ T\ \varphi && \text{automaton trigger}
\end{aligned}
$$

# Semantics of AFL

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

# Elementary Formulae of AFL

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 $b$, $\neg\psi$, $\psi_1 \wedge \psi_2$, $\mathsf{X}\psi$, $\psi_1 \mathsf{U} \psi_2$, their elementary formula sets are defined same as before.
  - $El(A \text{ abort! } b) = El(b) \cup \{\mathsf{X}(A^q \text{ abort! } b) \mid q \text{ is a state of } A\}$.
  - $El(A\ \mathsf{T}\psi) = El(\psi) \cup \{\mathsf{X}(A^q\ \mathsf{T}\psi) \mid q \text{ is a state of } A\}$.

# *Sat* function of AFL formulae

❖ *Sat* function of abort!

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

$=$

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

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

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 AFL formulae

- *Sat* function of T

$Sat(A^q \, \mathrm{T} \, \psi)$

$=$

$\{W \in Sat(\psi) \mid \delta(q, W \cap AP) \cap F \neq \emptyset\} \cup$

$\{W \subseteq El(\varphi) \mid \exists r \in \delta(q, W \cap AP)$, s.t. $X(A^r \, \mathrm{T} \, \psi) \in W\}$

$\pi, i \models A^q \mathrm{T} \psi$

iff

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

or $\pi, i \models X(A^r \mathrm{T} \psi)$ for some $r \in \delta(q, \pi(i))$

# Tableaux for AFL formulae

❖ 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 $P_j \subseteq Q_j$.

– $(\langle W, (P_1,\ldots,P_m)\rangle, \langle W', (P'_1,\ldots,P'_m)\rangle) \in \rho_\varphi$ iff

  - For each $X\psi \in El(\varphi)$, $W \in Sat(X\psi)$ iff $W' \in Sat(\psi)$;
  - For each $1 \leq j \leq m$, $((W,P_j),(W',P'_j)) \in \rho_{(A_j, \psi_j)}$.

– $I_\varphi = \{\langle W, (P_1,\ldots,P_m) \rangle \mid W \in Sat(\varphi)\}$.

– $L_\varphi(\langle W,(P_1,\ldots,P_m)\rangle) = W \cap AP$.

# Trigger Pair / Transition

Given an AFL formula $\varphi$, we say that $(A, \psi)$ is a trigger pair in $\varphi$, if there is some $q$, such that $A^q \mathrm{T} \psi$ 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 $Q_2 = \{q \mid W_2 \in Sat(A^q \mathrm{T} \psi)\}$.
- If $Q_1 \neq \emptyset$, then for each $q \in Q_1$,
  - either $W_1 \in Sat(\psi)$ and $\delta(q, W_1 \cap AP) \cap F \neq \emptyset$
  - or there is some $q' \in Q_2$, s.t. $q' \in \delta(q, W_1 \cap AP)$

# Trigger Transition Relation

- For each trigger pair $(A, \psi)$ and each state $q$ of $A$, and each $s \in Sat\ (A^q \mathsf{T} \psi)$, a fair path starting from $s$ must satisfy $A^q \mathsf{T} \psi$

# Fairness Constraints

- $\Omega_\varphi$ consists of three parts:
  - For each subformula $\psi_1 \mathsf{U} \psi_2$, create a fairness constraint $\{\langle W, (P_1,\ldots,P_m) \rangle \mid$
    $W \in Sat(\psi_2)$ or $W \notin Sat(\psi_1 \mathsf{U} \psi_2)\}$.
  - For each subformula $A^q$ abort! $b$, create a fairness constraint
    $\{\langle W, (P_1,\ldots,P_m) \rangle \mid$
    $W \in Sat(b)$ or $W \notin \bigcup_{r \in Q} Sat(A^r \text{ abort! } b)\}$.
  - For each trigger pair $(A_i, \psi_i)$, create a fairness constraint
    $\{\langle W, (P_1,\ldots,P_m) \rangle \mid P_i = \varnothing \}$.

# AFL Model Checking

- 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 $(M || T_{\varphi})$ 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

# BDD Encoding

- 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.

# BDD Encoding

- Given an AFL formula $\varphi$ with trigger pairs $(A_1,\psi_1)$, …,$(A_m,\psi_m)$ 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 $Sat(\psi)$.

# BDD Encoding

- The symbolic encoding of transition relation is the conjunction of the following issues:
  - For each $\mathsf{X}\psi \in El(\varphi)$, add a conjuct $u_{\mathsf{X}\psi} \Leftrightarrow f_{\psi}$.
  - For each $1 \leq i \leq m$, employ the following two conjuncts

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

If $Q_1 = \emptyset$, then $Q_2 = \{q \mid W_2 \in Sat(A^q \mathsf{T} \psi)\}$

# BDD Encoding

- For each subformula $\psi_1 \mathsf{U} \psi_2$,we add a fairness constraint encoding

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

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

$$f_b \vee \bigwedge_{q' \in Q} \neg f_{A^{q'} \text{abort!} b}$$

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

$$\bigwedge_{q \in Qi} \neg v_{(i,q)}.$$

# 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

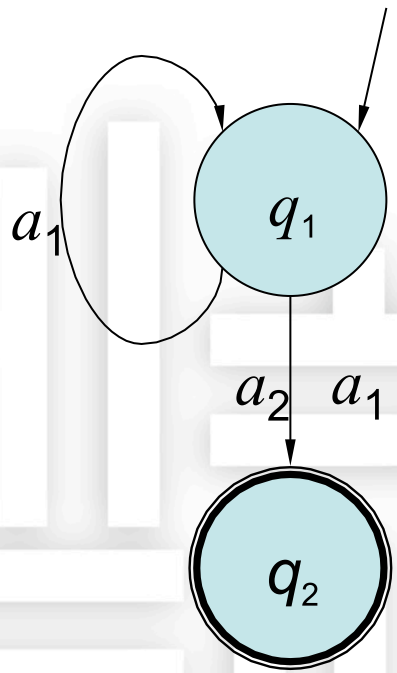
❖ From NuSMV to ENuSMV

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

# Automata Constructs

❖ Defining Automata constructs in ENuSMV




CONNECTIVE $A(a_1,a_2)$

STATES:

$> q_1, q_2 <$

TRANSITIONS$(q_1)$

case

$a_1$: $\{q_1,q_2\}$;

$a_2$: $q_2$;

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

# Philosopher Dining

| Feasibility | | | | | |
|---|---|---|---|---|---|
| #philosophers | # states | | #time (sec.) | | |
| | CTL | AFL | CTL | LTL | AFL |
| 6 | 1135 | 2270 | 0.020 | 0.046 | 0.046 |
| 7 | 3545 | 7090 | 0.037 | 0.122 | 0.124 |
| 8 | 11395 | 22790 | 0.070 | 0.310 | 0.311 |

# Philosopher Dining

| Liveness | | | | | |
|---|---|---|---|---|---|
| #philosopher | # states | | #time (sec.) | | |
| | CTL | AFL | CTL | LTL | AFL |
| 6 | 1135 | 4412 | 0.021 | 0.040 | 0.046 |
| 7 | 3545 | 14180 | 0.0367 | 0.073 | 0.074 |
| 8 | 11395 | 45580 | 0.070 | 0.150 | 0.149 |

# Verification of Non-star-free Properties

- Consider the model of "accumulative carry circuit ", which is a synchronous circuit composed 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.

# Accumulative Carry Circuit

```
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 & carray_in);
```

# Accumulative Carry Circuit

```
bit_0: counter(1);
… …
bit_i+1: counter(bit_i.output);
… …
bit_n: counter(bit_n-1.output);
```

# Accumulative Carry Circuit

- Consider the following properties:
  - Periodicity: bit_0 carries out at each odd moment.
  - Monitoring: once the waveform matches the pattern *true*;(bit_0.carry_out)$^*$, then the signal bit_1.carry_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

| Periodicity | | | Monitoring | | |
|---|---|---|---|---|---|
| #bits | # states | #time | #bits | # states | #time |
| 7 | 66560 | 0.011 | 7 | 8320 | 0.009 |
| 8 | 132096 | 0.012 | 8 | 16512 | 0.011 |
| 9 | 263168 | 0.019 | 9 | 32896 | 0.018 |
| 10 | 525312 | 0.024 | 10 | 65664 | 0.021 |

# Other Experiments

- To test the scalability, we have done DME (Distributed Mutual Exclusive) circuit experiment. Results show that ENuSMV can handle models with more than $5\times10^{21}$ 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?