Model Checking VI
Linear-Time Temporal Logic

Edmund M. Clarke, Jr.
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Model Checking for LTL

- Reduction of LTL model checking to CTL model checking with fairness constraints
- *Symbolic* LTL model checking algorithm
- Extension of SMV to permit LTL specifications

- O. Lichtenstein and A. Pnueli. Checking that finite state concurrent programs satisfy their linear specification. In *Proceedings of the Twelfth Annual ACM Symposium on Principles of Programming Languages*, January 1985.
- E.M. Clarke, O. Grumberg and K. Hamaguchi. Another Look at LTL model checking. In *Proceedings of the 1994 Conference on Computer-Aided Verification*, June 1994.

Motivation

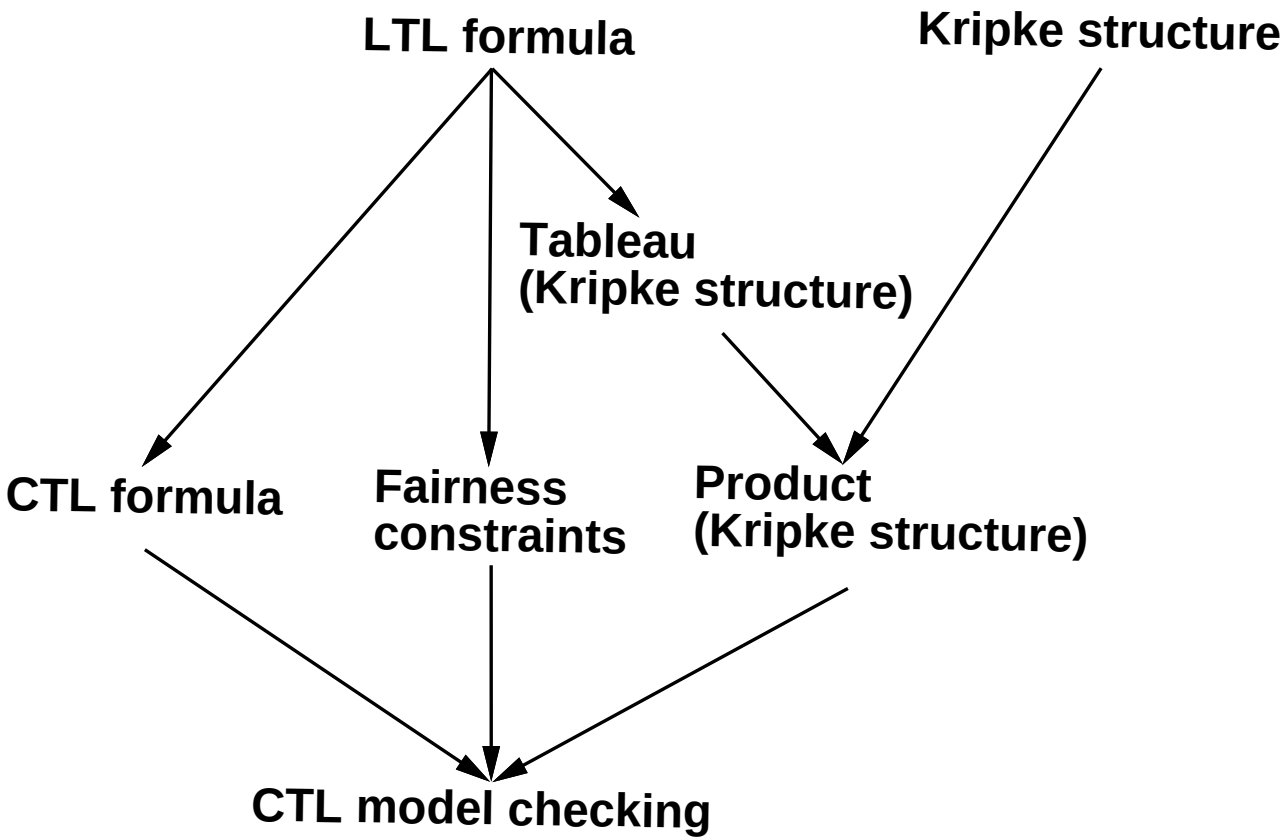
- Succinct and intuitive descriptions
  - No path quantifiers
  - Paths rather than trees
- Expressiveness
  - Some properties such as $\mathbf{FG}\, p$ cannot be expressed in CTL.

Review of LTL Syntax

Syntax

- LTL formula:
  $\mathbf{A}\, f$ (f is a path formula).
- Path formulas:
  - Propositional operators: $\neg p$ and $p \vee q$
  - Temporal operators: $\mathbf{X}\, p$ and $p\, \mathbf{U}\, q$

Basic Idea for Reduction



Major Steps in Reduction

1. Translate the given LTL formula $\mathbf{A}\, f$ to:
   - Tableau (Kripke structure) $T = (S_T, R_T, L_T)$: Includes every path that satisfies $\neg f$.
   - Fairness constraints $\mathcal{F}$: Guarantee that every eventuality $g\, \mathbf{U}\, h$ is ultimately filfilled.
   - CTL formula ψ: Guarantees that no state is the start of a path that satisfies $\neg f$.
2. Generate the Product P of M and T
3. Perform CTL model checking of ψ in P under $\mathcal{F}$.

States of Tableau

S_T is $\mathcal{P}(el(p))$, i.e, the power set of *elementary* formulas of f.

- $el(p) = \{p\}$ if $p \in AP$.
- $el(\neg g) = el(g)$.
- $el(g \vee h) = el(g) \cup el(h)$.
- $el(\mathbf{X}\, g) = \{\mathbf{X}\, g\} \cup el(g)$.
- $el(g\, \mathbf{U}\, h) = \{\mathbf{X}(g\, \mathbf{U}\, h)\} \cup el(g) \cup el(h)$.

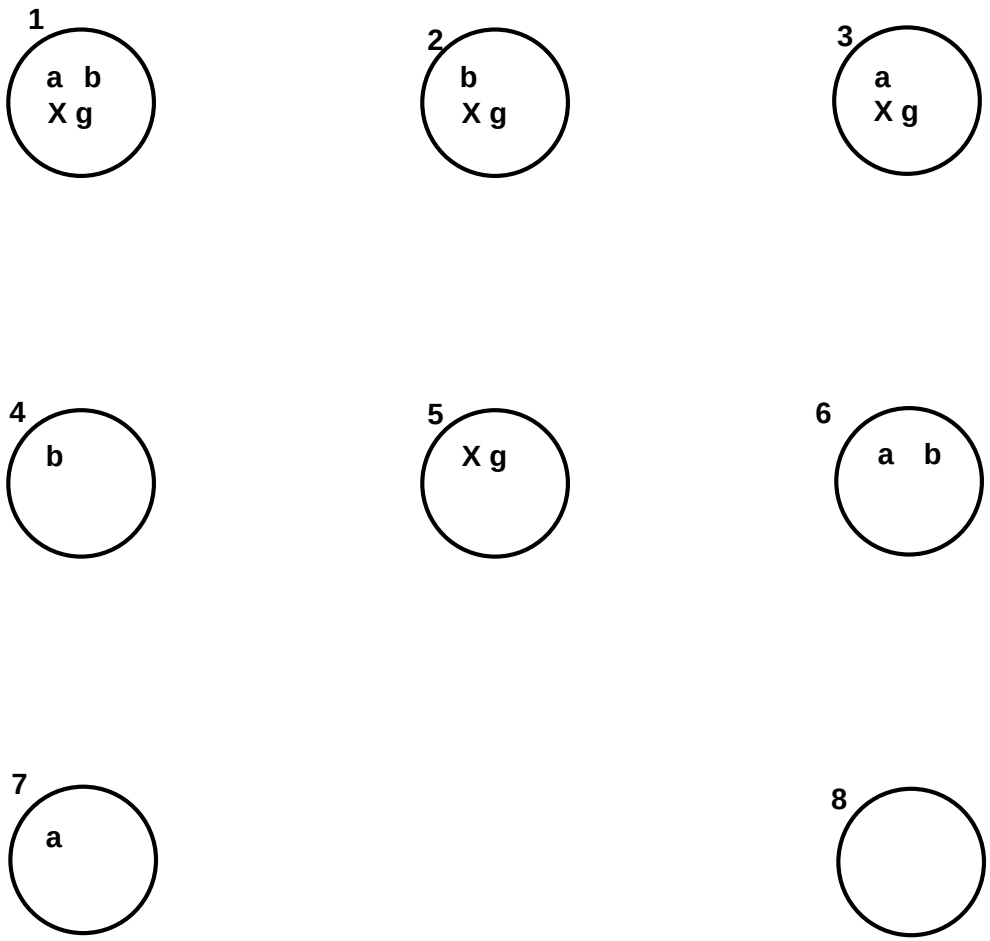
Example.

$$el(a\, \mathbf{U}\, b) = \{a, b, \mathbf{X}(a\, \mathbf{U}\, b)\}$$

$$el(a\, \mathbf{U}\, (\mathbf{X}\, b)) = \{a, b, \mathbf{X}\, b, \mathbf{X}(a\, \mathbf{U}\, (\mathbf{X}\, b))\}$$

Simple Example

States in tableau T for $g = a \mathbf{U} b$:



Transition Relation for Tableau

Additional function sat:

$sat(g)$ will be the set of states that satisfy g.

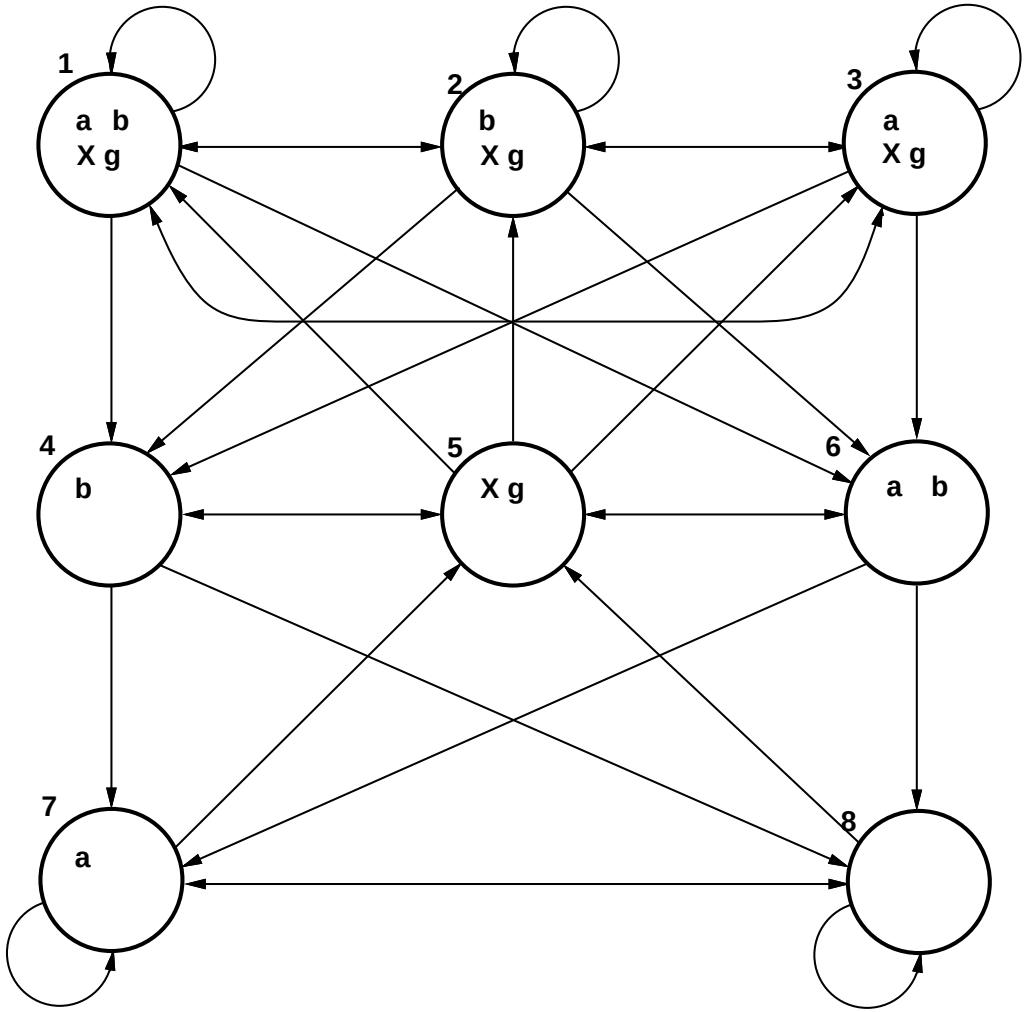
- $sat(g) = \{\sigma \mid g \in \sigma\}$ where $g \in el(f)$.
- $sat(\neg g) = \{\sigma \mid \sigma \notin sat(g)\}$.
- $sat(g \vee h) = sat(g) \cup sat(h)$.
- $sat(g \,\mathbf{U}\, h) = sat(h) \cup \big(sat(g) \cap sat(\mathbf{X}(g \,\mathbf{U}\, h))\big)$.

Transition relation R_T:

$$R_T(\sigma, \sigma') = \bigwedge_{\mathbf{X}g \in el(f)} \sigma \in sat(\mathbf{X}\, g) \Leftrightarrow \sigma' \in sat(g).$$

Simple Example (Cont.)

Tableau T for $g = a \mathbf{U} b$:



Fairness Constraints for Reduction

We must guarantee that every eventuality is actually fulfilled. For this purpose we use the following fairness constraints.

Fairness Constraints $\mathcal{F}$:

$$\{sat(\neg(g \,\mathbf{U}\, h) \vee h) \mid g \,\mathbf{U}\, h \text{ occurs in } f\}.$$

Correctness of Reduction

Theorem: If $M, \pi' \models \neg f$ for some M and π',
then there exists a path π in T such that:

- π is a fair path.
- The initial state of π is in $sat(\neg f)$.

Theorem: $M, \sigma' \models \mathbf{A}\, f$ if and only if
there is NO state (σ, σ') in P such that:

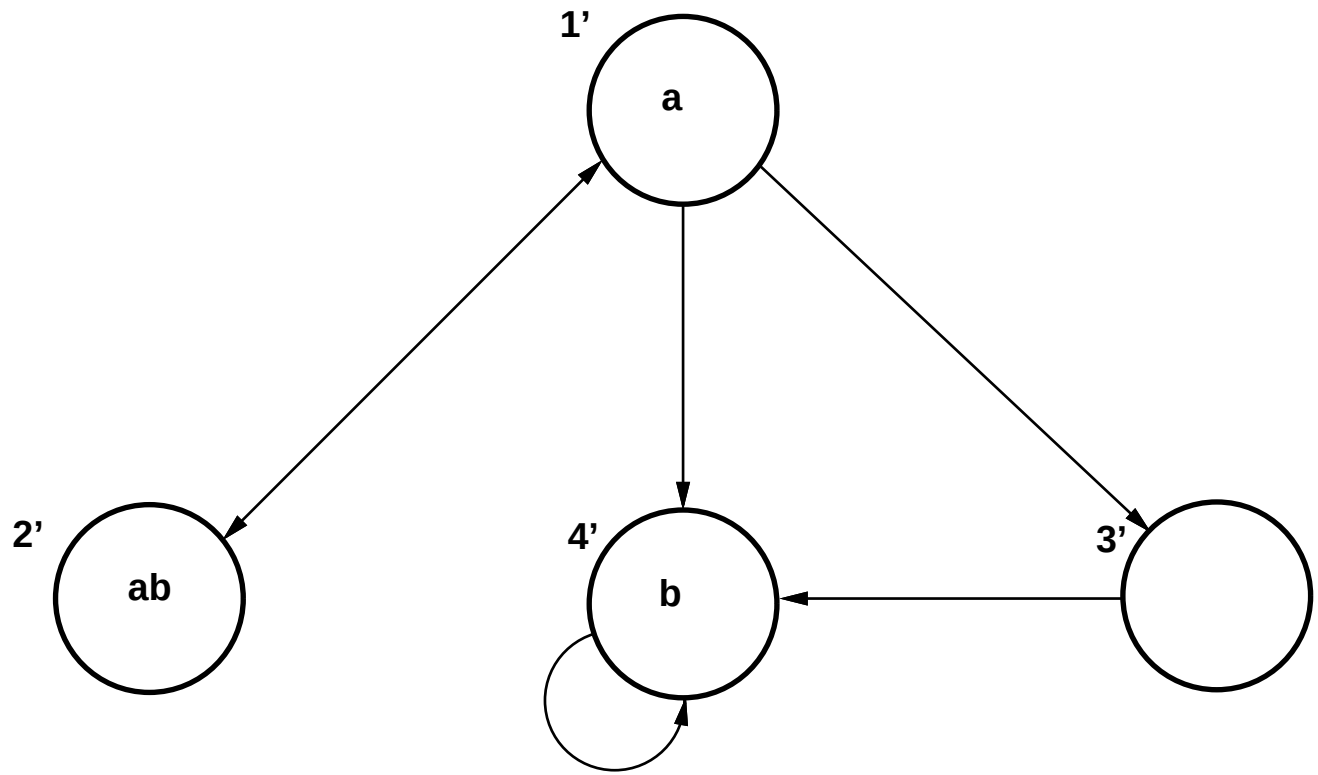
- $P, (\sigma, \sigma') \models \mathbf{EG}\, true$ under the fairness constraints.
- $(\sigma, \sigma') \in sat(\neg f)$.

Sufficient to check the CTL formula ψ:

$$\neg(\mathbf{EG}\, true \;\&\; Sat_{\neg f})$$

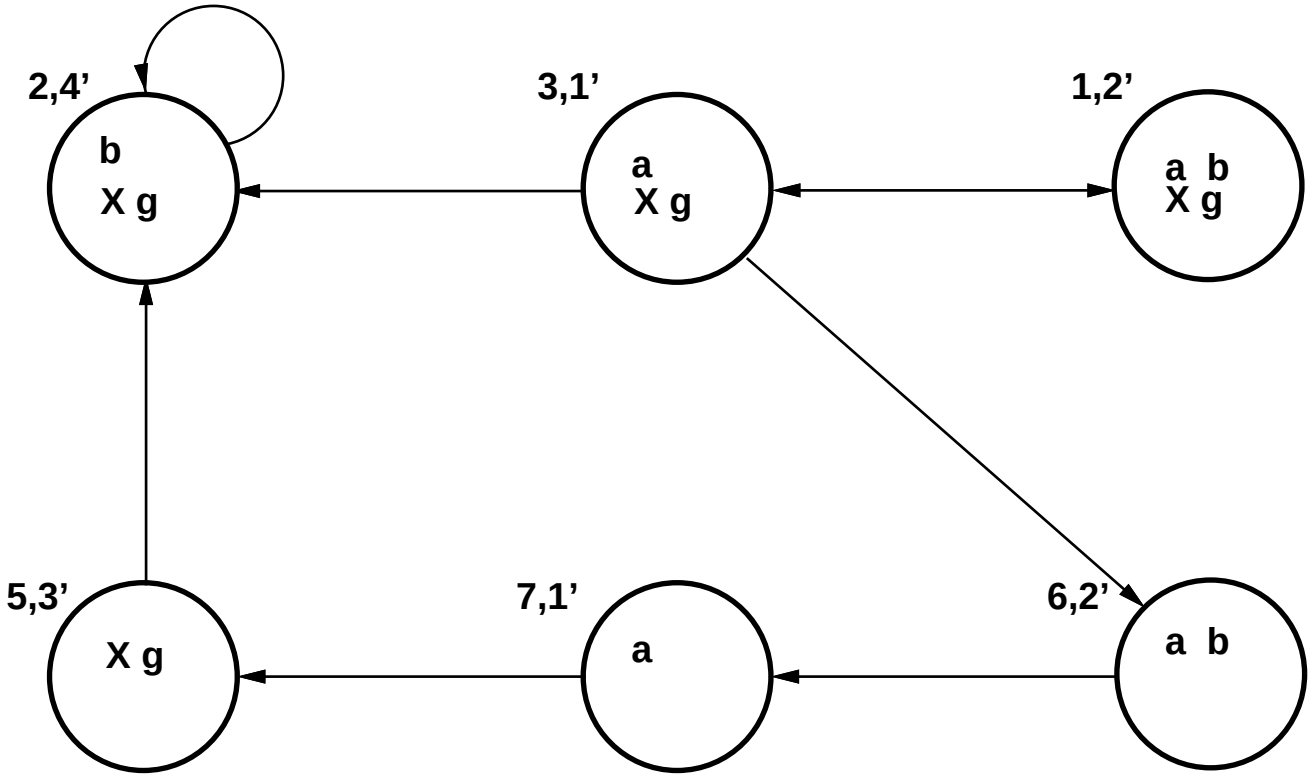
Simple Example (Cont.)

The Kripke structure M:



Simple Example (Cont.)

Product P of the structure M and the tableau T:



Simple Example (Cont.)

- Fairness Constraint $\mathcal{F}$:
  $a \,\mathbf{U}\, b$ is eventually fulfilled.

- CTL formula ψ:

$$\neg(\mathbf{EG}\, true \;\&\; Sat_{\neg(a \mathbf{U} b)})$$

LTL Model Checking for SMV

We have developed a translator that extends SMV to permit LTL specifications.

The translator replaces a given LTL formula with SMV code for

- a tableau,
- fairness constraints and
- a CTL formula.

The tableau description is implicit !!

Simple Example (Cont.)

```
MODULE main  -- simple program

VAR
a: boolean;
b: boolean;

TRANS   ( a & !b) -> next(!(a & !b))
TRANS   ( a &  b) -> next(a & !b)
TRANS   (!a &  b) -> next(!a & b)
TRANS   (!a & !b) -> next(!a & b)

SPEC    A[a U b]
```

Translation

```
-- Kripke structure
MODULE
                    ⋮
MODULE              main
                    ⋮
```

```
-- LTL formula
SPEC                A f
```

An SMV program

Translation

```
-- Tableau for f
VAR         -- new variables
            EL_{X g_1} : boolean;
            :
            EL_{X g_N} : boolean;

DEFINE      -- characteristic function
            S_{h_1} := ···;
            :
            S_{h_M} := ···;

TRANS       -- transition relation
            ( S_{X g_1} = next (S_{g_1}) ) &
            :
            ( S_{X g_N} = next (S_{g_N}) )

-- fairness constraints
FAIRNESS    !S_{g'_1 U h'_1} | S_{h'_1}
            :
FAIRNESS    !S_{g'_3 U h'_3} | S_{h'_3}

-- new specification
SPEC        !(S_{¬f} & EG true)
```

Translator output for SMV program

Simple Example (Cont.)

Result of translation for simple example:

```
MODULE main  -- simple program
VAR        a: boolean;
           b: boolean;
TRANS      ( a & !b) -> next(!(a & !b))
TRANS      ( a &  b) -> next(a & !b)
TRANS      (!a &  b) -> next(!a & b)
TRANS      (!a & !b) -> next(!a & b)
VAR        EL_X_a_U_b : boolean;
DEFINE     S_a              := a;
           S_b              := b;
           S_X_a_U_b        := EL_X_a_U_b;
           S_a_U_b := S_b | (S_a & S_X_a_U_b);
           S_NOT_a_U_b      := !S_a_U_b;
TRANS      S_X_a_U_b = next(S_a_U_b)
FAIRNESS   !S_a_U_b | b
SPEC       !(S_NOT_a_U_b & EG true)
```

Experimental Results

- Distributed mutual exclusion (DME) circuit
  - Speed-independent token ring composed of identical cells
  - Each gate is modeled as a non-deterministic finite-state machine.
- Specifications
  1. (*Safety*) No two users are acknowledged simultaneously.
  2. (*Liveness*) All requests are eventually acknowledged.

- Results (time and space) : Comparison with CTL model checking
  - (*Safety*) Within 10% increase
  - (*Liveness*) Within 1.5-3 times increase (2 times for large circuits)

Experimental Results (Cont.)

#cell	#nodes		#time(sec)		trans.		#reachable states	
	CTL	LTL	CTL	LTL	CTL	LTL	CTL	LTL
3	11326	11362	17.9	20.5	2778	2781	6579	13158
4	13458	15357	47.5	49.4	4757	4760	75172	150344
5	22321	22348	100.5	104.4	6760	6763	802425	1.60485e+06
6	25869	27318	182.3	193.6	8763	8766	8.2166e+06	1.64332e+07
7	28413	33310	326.4	329.3	10766	10769	8.1784e+07	1.63568e+08
8	44322	44369	509.2	526.3	12769	12772	7.97393e+08	1.59479e+09
9	49702	49755	794.0	794.8	14772	14775	7.65302e+09	1.53060e+10
10	55082	55141	1125.2	1362.7	16775	16778	7.30144e+10	1.46029e+11

Table: Safety specification for the DME circuit

#cell	#nodes		#time(sec)		trans.		#reachable states	
	CTL	LTL	CTL	LTL	CTL	LTL	CTL	LTL
3	12721	33940	426.1	1260.5	2778	3004	6579	26316
4	26541	72029	2553.2	6096.7	4757	4983	75172	300688
5	47346	120299	9623.1	21950.1	6760	6986	802425	3.2097e+06
6	92080	183043	36995.3	66502.5	8763	8989	8.2166e+06	3.28664e+07
7	163867	263380	97807.1	191990.0	10766	10992	8.1784e+07	3.27136e+08

Table: Liveness specification for the DME circuit

Experimental Results (Cont.)

- Synchronous bus arbiter
  - Daisy chain circuit composed of identical cells
  - Each gate is modeled by a deterministic machine
- Specifications
  1. (*Safety*) No two users are acknowledged simultaneously.
  2. (*Liveness*) All requests are eventually acknowledged.

- Results (time and space) : Comparison with CTL model checking
  - (*Safety*) Within 1.5 times increase
  - (*Liveness*) Within 1.5-2 times increase

Experimental Results (Cont.)

#cell	#nodes		#time(sec)		trans.		#reachable states	
	CTL	LTL	CTL	LTL	CTL	LTL	CTL	LTL
5	987	1913	0.11	0.15	144	318	10240	20480
6	1383	2628	0.13	0.18	176	418	49152	98304
7	1842	3424	0.16	0.21	208	518	229376	458752
8	2364	4301	0.16	0.26	240	618	1.04858e+06	2.09715e+06
9	2949	5259	0.16	0.33	272	718	4.71859e+06	9.43718e+06
10	3597	6298	0.21	0.33	304	818	2.09715e+07	4.19430e+07
11	4308	7418	0.21	0.41	336	918	9.22747e+07	1.84549e+08
12	5082	8619	0.31	0.45	368	1018	4.02653e+08	8.05306e+08

Table: Safety specification for the sync. arbiter

#cell	#nodes		#time(sec)		trans.		#reachable states	
	CTL	LTL	CTL	LTL	CTL	LTL	CTL	LTL
5	2155	4254	0.38	0.43	144	258	10240	40960
6	2867	5483	0.43	0.48	176	320	49152	196608
7	3667	6820	0.48	0.61	208	382	229376	917504
8	4555	8266	0.53	0.81	240	444	1.04858e+06	4.1943e+06
9	5531	9821	0.71	1.01	272	506	4.71859e+06	1.88744e+07
10	6595	10000	0.83	1.23	304	568	2.09715e+07	8.38861e+07
11	7747	10001	1.00	1.46	336	630	9.22747e+07	3.69099e+08
12	8987	10052	1.16	1.71	368	692	4.02653e+08	1.61061e+09

Table: Liveness specification for the sync. arbiter