CS 745 / ECE 725
Computer Aided Verification

Lecture 4: Introduction to Model Checking

Jo Atlee

DC 2337, jmatlee@uwaterloo.ca

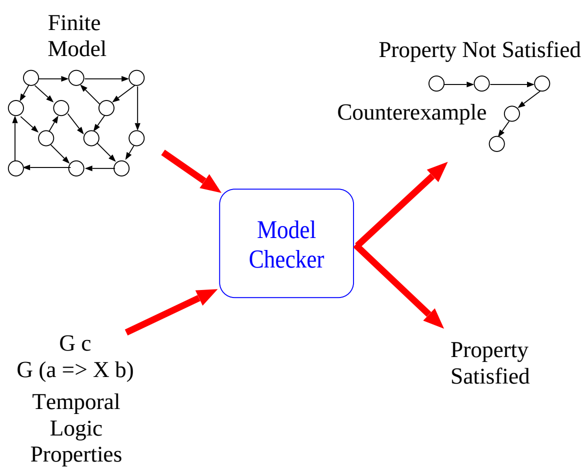
Office Hours: Mon 1:00-2:00, Wed 1:00-2:00

http://www.student.cs.uwaterloo.ca/~cs745

Today's Agenda

- Model checking overview
- Linear temporal logic
  - Handshaking example handout
- Kripke structures
- Computation tree logic (CTL)

Model Checking



Model Checking

$$\mathcal{M} \models \phi$$

- model $\mathcal{M}$ is a Kripke structure (labelled state-transition graph) that specifies how variables may change value
- ϕ is a temporal logic property
- $\models$ relates a model $\mathcal{M}$ and property ϕ, informally saying that all paths through $\mathcal{M}$ satisfy ϕ

There are different temporal logics. For each temporal logic, we will define what $\models$ means.

Models and Entailment

The meanings of $\models$ in model checking are comparable to its meanings in propositional and predicate logic [HR04]:

1. $\psi \models \phi$ relates two formulas, saying that forall v (i.e., for possible variable values), if $v(\psi) =$ T then $v(\phi) =$ T. This is called semantic entailment.
2. Each state in a Kripke structure $\mathcal{M}$ represents a valuation, and paths through $\mathcal{M}$ represent sequences of valuations. $\mathcal{M} \models \phi$ says that all sequences of valuations allowed by $\mathcal{M}$ satisfy ϕ. This is called satisfaction relationship.

Model Checking vs. Testing

Both check execution paths of a model / program.

- Testing/Simulation only checks the behaviour on selected inputs.
- Model Checking checks all behaviours, but checks only specified properties.

Historical Perspective

- Burstall, Pnueli and others proposed using temporal logic (TL) for reasoning about programs (early 1970's)
- Pnueli was the first to use TL for reasoning about concurrency [Pnu77] (1977)
- Clarke and Emerson [CE81] in the early 1980's developed a way to automate temporal logic reasoning (CTL); Quielle and Sifakis [QS81] also gave a model checking algorithm at this time.

Historical Perspective

- language containment approach to model checking
  - SPIN - Holzmann [Hol]
  - COSPAN/Formalcheck – Kurshan [Kur94]
- symbolic model checking (implicit representation of the state space) – McMillan, 1987 [BCM+92]
- lots and lots of work on handling the state space explosion problem

Another approach to model checking is symbolic trajectory evaluation (STE), developed by Bryant and Seger [SB95]. STE may be a topic for a Paper Presentation.

Temporal Logic

Temporal logic is used to describe changes in values over time.

We will discuss propositional temporal logic.

So we now consider that propositions can have different truth values at different times.

We can use all the regular logical connectives to create statements in propositional logic.

A temporal logic formula is considered relative to an initial state ("now").

Linear Temporal Logic (LTL) Syntax

Temporal logics use temporal operators. Linear temporal logic uses the following operators:

Symbol	Alternate Symbol	Informal Meaning
X	$\bigcirc$	Next
F	$\Diamond$	Eventually (in the future)
G	$\Box$	Always (globally, henceforth)
U	$\mathcal{U}$	Strong until
W	$\mathcal{W}$	Weak until

LTL Syntax

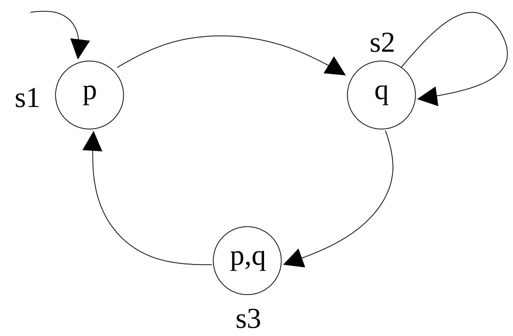
If p is an atomic proposition, and f_1 and f_2 are LTL formulae, then the set of LTL formulae consists of:

1. p
2. $\neg f_1$, $f_1 \wedge f_2$, $f_1 \vee f_2$, $f_1 \Rightarrow f_2$
3. **X**f_1, **G**f_1, **F**f_1, f_1 **U** f_2, f_1 **W** f_2

Brackets are used as necessary.

LTL Semantics

A temporal formula f is evaluated with respect to a Kripke structure: a state-transition graph whose states are labelled with propositional variables:



Kripke Structures

Let AP be a set of atomic propositions. A Kripke structure $\mathcal{M}$ over AP is a four tuple $\mathcal{M} = (S, S_0, R, L)$ where

1. S is a finite set of states.
2. $S_0 \subseteq S$ is the set of initial states.
3. $R \subseteq S \times S$ is a transition relation that must be total, that is $\forall s \in S.\ \exists s'.\ R(s, s')$.
4. $L : S \rightarrow 2^{\text{AP}}$ is a function that labels each state with the set of atomic propositions true in that state.

Note: there are no labels on the transition arrows.

Labelling Function

$$L : S \rightarrow 2^{\text{AP}}$$

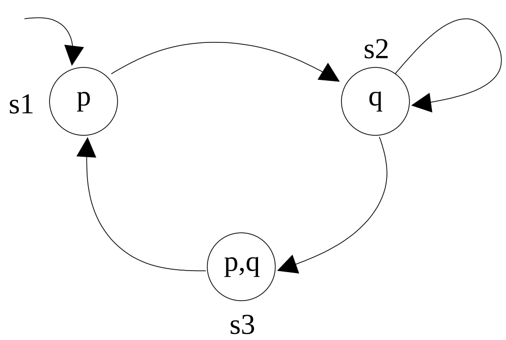
AP is the set of atomic propositions. For example, AP might be $\{p, q, r\}$.

2^{AP} is the power set of the set of atomic propositions.

$$2^{\{p,q,r\}} = \{\emptyset, \{p\}, \{q\}, \{r\}, \{p,q\}, \{p,r\}, \{q,r\}, \{p,q,r\}\}$$

L is a function that takes a state as an argument and returns the set of propositions that are true in that state.

Labelling Function Example

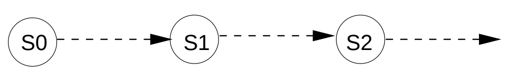


What is the labelling function in this example?

Paths

LTL formulas are evaluated relative to paths (i.e., an LTL formula is true or false relative to a sequence of valuations).

For a Kripke structure, a path from a state s_0 is an infinite sequence $\pi = s_0 s_1 s_2 \ldots$ such that $\forall i. R(s_i, s_{i+1})$.

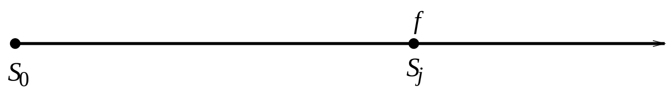


We use π_i to denote the suffix of π starting at s_i.

LTL Semantics

An ordinary predicate logic formula (a state formula) is evaluated with respect to a single state.

$$\mathcal{M}, \pi_i \models f \quad \text{iff} \quad f \text{ is true in state } s_j \text{ of } \pi \text{ in } \mathcal{M}$$



Normally, formulae are evaluated with respect to the initial state of the model:

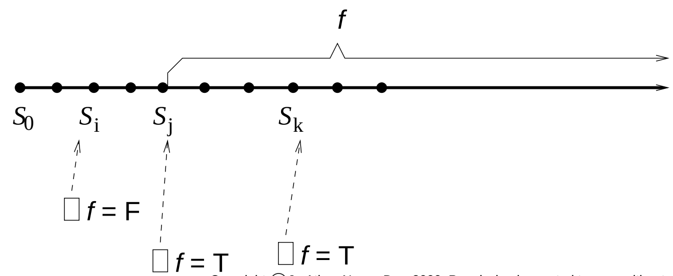
$$\mathcal{M}, \pi \models f \quad \equiv \quad \mathcal{M}, \pi_0 \models f$$
$$\mathcal{M} \models f \quad \text{iff} \quad f \text{ is true in state } s_0 \text{ of all paths } \pi \text{ in } \mathcal{M}$$

Henceforth (or Globally or Always)

$$\mathbf{G} f = \begin{cases} T & \text{if } f \text{ is true in the current and \underline{all} future states} \\ F & \text{otherwise} \end{cases}$$

where f is any LTL formula.

$$\pi_j \models \mathbf{G} f \quad \text{iff } \forall i \bullet j \leq i \Rightarrow \pi_i \models f$$



Eventually

$$\mathbf{F} f = \begin{cases} T & \text{if } f \text{ is true in the current or \underline{some} future state} \\ F & \text{otherwise} \end{cases}$$

$$\pi_j \models \mathbf{F} f \quad \text{iff} \quad \exists i \bullet j \leq i \wedge \pi_i \models f$$

Examples

1. The number of subway riders is always less than or equal to the number of subway tokens inserted into turnstile coinslot.

2. If a rider pushes the turnstile then eventually the rider will enter the subway station.

3. Whenever a rider pushes the turnstile then eventually the rider will enter the subway station.

Examples

$\mathbf{F}(\mathbf{G}f)$

$\mathbf{G}(\mathbf{F}f)$

Next

$$\mathbf{X}f = \begin{cases} T & \text{if } f \text{ is true in the \underline{next} state} \\ F & \text{otherwise} \end{cases}$$

$\pi_j \models \mathbf{X}f \quad \text{iff} \quad \pi_{j+1} \models f$

Until

$$f\,\mathbf{U}\,g = \begin{cases} T & \text{if } g \text{ is eventually true, and} \\ & f \text{ is true until } g \text{ is true} \\ F & \text{otherwise} \end{cases}$$

$\pi_j \models f\,\mathbf{U}\,g \quad \text{iff}$

$\exists k \bullet k \geq j \wedge (\pi_k \models g) \wedge \forall i \bullet j \leq i < k \Rightarrow (\pi_i \models f)$

Unless

Unless is like Until, without the guarantee that g might happen. Unless is also called "weak until".

$$f\,\mathbf{W}\,g = \begin{cases} T & \text{if } f \text{ holds indefinitely or } f \text{ holds until } g \text{ holds} \\ F & \text{otherwise} \end{cases}$$

$f\,\mathbf{W}\,g \quad \text{iff} \quad (f\,\mathbf{U}\,g) \vee \mathbf{G}f$

It rains unless I take my umbrella

$r\mathbf{W}u$ (when I get my umbrella it might or might not rain)

More Examples: Elevator

Let AP =
moving
door_open, door_closed
req[1..5]

1. The elevator shall never move with its doors open.

2. The elevator shall not keep its doors open indefinitely.

3. The elevator does not move before a request is made

LTL Semantics

$\mathcal{M}, \pi \models g$ means the LTL formula g holds on path $\pi = s_0 s_1 s_2 \ldots$ in the Kripke structure $\mathcal{M} = (S, S_0, R, L)$. The relation $\models$ is defined inductively as follows:

$\mathcal{M}, \pi \models p$	iff	$p \in L(s_0)$ (where p is an atomic proposition)
$\mathcal{M}, \pi \models \neg g$	iff	$\mathcal{M}, \pi \not\models g$
$\mathcal{M}, \pi \models g_1 \vee g_2$	iff	$\mathcal{M}, \pi \models g_1$ or $\mathcal{M}, \pi \models g_2$
$\mathcal{M}, \pi \models g_1 \wedge g_2$	iff	$\mathcal{M}, \pi \models g_1$ and $\mathcal{M}, \pi \models g_2$
$\mathcal{M}, \pi \models \mathbf{X}\ g$	iff	$\mathcal{M}, \pi_1 \models g$
$\mathcal{M}, \pi \models \mathbf{G}\ g$	iff	$\forall i \geq 0,\ \mathcal{M}, \pi_i \models g$
$\mathcal{M}, \pi \models \mathbf{F}\ g$	iff	$\exists i \geq 0,\ \mathcal{M}, \pi_i \models g$
$\mathcal{M}, \pi \models g_1\ \mathbf{U}\ g_2$	iff	$\exists i \geq 0,\ \mathcal{M}, \pi_i \models g_2$ and $\forall j.\ 0 \leq j < i \Rightarrow \mathcal{M}, \pi_j \models g_1$

LTL Theorems

Some LTL formulae are true in all models. These are called theorems or tautologies.

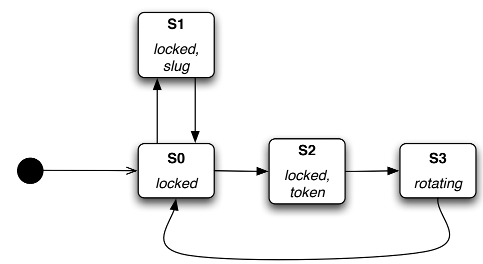
$(\mathbf{X}p) \Rightarrow (\mathbf{F}p)$

$(f\ \mathbf{U}\ g) \Rightarrow \mathbf{F}g$

$g \Rightarrow (f \mathbf{U} g)$

$(f \wedge \mathbf{X}g) \Rightarrow (f \mathbf{U} g)$

Evaluating LTL Properties



True or false?
$\mathcal{M} \models \mathbf{F}(rotating)$
$\mathcal{M} \models \mathbf{G}(locked \Rightarrow (locked\ \mathbf{U}\ token))$
$\mathcal{M} \models \mathbf{G}((locked \wedge token) \Rightarrow \mathbf{X}(\neg locked))$

LTL Semantics

We just defined $\mathcal{M}, \pi \models g$, where π is a path.

An LTL formula g is satisfied in a state s of a model $\mathcal{M}$ if g is satisfied on every path starting at s.

How do we say there is some path where a formula is true?

For example, “from any state, there is always some way to get to a state where p is true”.

In LTL we can’t say this property. We turn now to another temporal logic: CTL.

LTL is conceptually simpler than CTL because we only have to think about one path at a time.

Linear and Branching Views

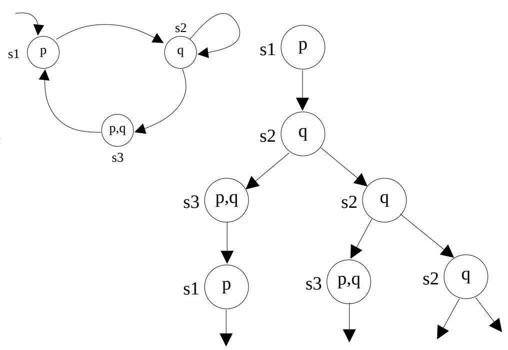
There are two ways to think about the computations of a system:

- linear-time: there is a single time line and a computation is the sequence of states that the system goes through in that time.
- branching-time: the possible computations of the system are described by a tree

LTL uses linear-time and CTL uses branching-time.

Computation Trees

A computation tree is the unwinding of a Kripke structure starting from some state at its root.



Computation Tree Logic (CTL)

In CTL, there are the temporal operators of LTL, but there are also path quantifiers. These path quantifiers are used to describe the branching structure of a computation tree.

There are two path quantifiers:

- **A** means for all computation paths
- **E** means for some computation paths

These are used to describe the behaviour of the system from a particular state.

In CTL, we talk about a formula being true of a state rather than a path. (Then we check that it is true for all initial states of the system.)

CTL Syntax

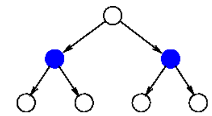
If p is an atomic proposition, and f_1 and f_2 are CTL formulae, then the set of CTL formulae consists of:

1. p
2. $\neg f_1$, $f_1 \wedge f_2$, $f_1 \vee f_2$, $f_1 \Rightarrow f_2$
3. **AX**f_1, **EX**f_1
4. **AG**f_1, **EG**f_1
5. **AF**f_1, **EF**f_1
6. **A**$[f_1$**U**$f_2]$, **E**$[f_1$**U**$f_2]$

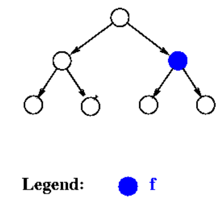
Note that the path quantifiers and temporal operators are always paired together.

Meaning of CTL Formulae

AXf if on all paths starting at state s, f holds in the next state.

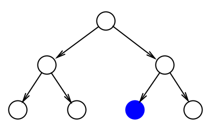


EXf if there exists a path starting at state s on which f holds at the next state.

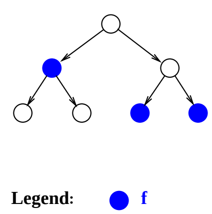


Meaning of CTL Formulae

EFf if f is reachable (i.e., if there exists a path starting at state s, on which f holds in some future state).

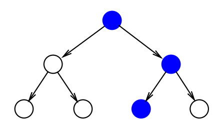


AFf if f is inevitable (i.e., if on all paths that start at state s, f holds in some future state).

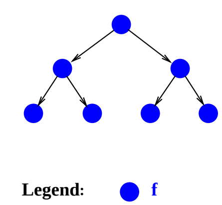


Meaning of CTL Formulae

EGf if there exists a path starting at state s, on which f holds globally.



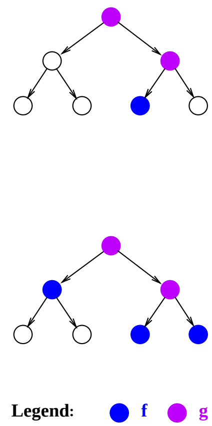
AGf if f is invariant (i.e., if on all paths that start at state s, f holds globally).



Meaning of CTL Formulae

E$[g$ **U** $f]$ if there exists a path starting at state s, on which g holds until f eventually holds.

A$[g$ **U** $f]$ if on all paths that start at state s, g holds until f eventually holds.



Examples of CTL Formulae

"Two processes cannot be in their critical section simultaneously."

"Every request will eventually be granted."

"There are requests."

"A request that is made infinitely often is eventually granted."

Examples of CTL Formulae

"Whatever happens, the system will eventually be permanently "deadlocked"

It is always possible to get to a state where restart is true.

Semantics of CTL

CTL formulae are evaluated with respect to a Kripke structure $\mathcal{M} = (S, S_0, R, L)$ and a state s (recall $\pi = s_0 s_1 s_2 \ldots$).

$$\begin{array}{rcl}
\mathcal{M}, s \models p & \text{iff} & p \in L(s) \text{ where } p \text{ is an atomic proposition} \\
\mathcal{M}, s \models \neg g & \text{iff} & \mathcal{M}, s \not\models g \\
\mathcal{M}, s \models f_1 \vee f_2 & \text{iff} & \mathcal{M}, s \models f_1 \text{ or } \mathcal{M}, s \models f_2 \\
\mathcal{M}, s \models f_1 \wedge f_2 & \text{iff} & \mathcal{M}, s \models f_1 \text{ and } \mathcal{M}, s \models f_2 \\
\mathcal{M}, s_0 \models \textbf{EX} f & \text{iff} & \exists\, s_1 \text{ such that } R(s, s_1) \text{ and } \mathcal{M}, s_1 \models f \\
\mathcal{M}, s_0 \models \textbf{AX} f & \text{iff} & \forall\, s_1 \text{ such that } R(s, s_1) \text{ we have } \mathcal{M}, s_1 \models f
\end{array}$$

Semantics of CTL

$\mathcal{M}, s_0 \models$ **EF**f iff in **some path** $s, s_1, s_2, ...$ such that $\forall i : R(s_i, s_{i+1})$ there is **some state** s_j where $\mathcal{M}, s_j \models f$

$\mathcal{M}, s_0 \models$ **AF** f iff in **all paths** $s, s_1, s_2, ...$ such that $\forall i : R(s_i, s_{i+1})$ there is **some state** s_j where $\mathcal{M}, s_j \models f$

$\mathcal{M}, s_0 \models$ **EG** f iff in **some path** $s, s_1, s_2, ...$ such that $\forall i : R(s_i, s_{i+1})$, for **all states** s_j on the path, we have $\mathcal{M}, s_j \models f$

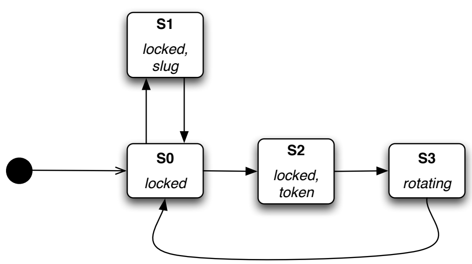
$\mathcal{M}, s_0 \models$ **AG** f iff in **all paths** $s, s_1, s_2, ...$ such that $\forall i : R(s_i, s_{i+1})$, for **all states** s_j on the path, we have $\mathcal{M}, s_j \models f$

Semantics of CTL

$\mathcal{M}, s_0 \models$ **E**$[f_1$ **U** $f_2]$ iff in **some path** $s, s_1, s_2, ...$ such that $\forall i : R(s_i, s_{i+1})$, there is **some state** s_j where $\mathcal{M}, s_j \models f_2$ and $\forall k . 0 \leq k < j$, we have $\mathcal{M}, s_j \models f_1$

$\mathcal{M}, s_0 \models$ **A**$[f_1$ **U** $f_2]$ iff in **all paths** $s, s_1, s_2, ...$ such that $\forall i : R(s_i, s_{i+1})$, there is **some state** s_j where $\mathcal{M}, s_j \models f_2$ and $\forall k . 0 \leq k < j$, we have $\mathcal{M}, s_j \models f_1$

Evaluating CTL Properties



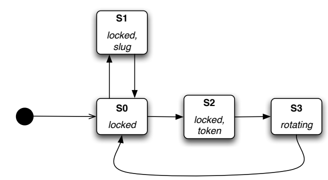
True or false?

$\mathcal{M} \models$ **AF**$(rotating)$

$\mathcal{M} \models$ **EG**$(locked)$

$\mathcal{M} \models$ **EG**$($**AF**$(\neg locked))$

LTL vs CTL



Can we write the following in LTL?

- $\mathcal{M} \models$ **EG**$(locked)$
- $\mathcal{M} \models$ **EG**$($**AF**$(\neg locked))$

LTL vs CTL

There are also some LTL properties that cannot be expressed in CTL. For example [HR04]:

It is possible in LTL to refer to a property that holds on a subset of paths. In LTL, we can say that "all paths that have a p state (where AP p holds) also have a q state."

$$\mathbf{F}p \Rightarrow \mathbf{F}q$$

We cannot state this in CTL, because the path formula we have, **AF** and **EF**, refer to all paths. The "corresponding" CTL formula means something quite different:

$$\mathbf{AF}p \Rightarrow \mathbf{AF}q$$

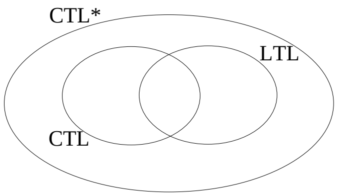
CTL*

CTL* is another temporal logic, that combines CTL with LTL. In CTL*, we can nest temporal operators and Boolean connectives before applying the path quantifiers.

In CTL*, we can write properties such as:

$$\mathbf{A}(\mathbf{X}p \vee \mathbf{XX}p)$$

$$\mathbf{E}(\mathbf{GF}p)$$



See Huth and Ryan [HR04] or Clarke et al. [CGP99] for further discussion of CTL*.

Specification Patterns

Further examples demonstrating how natural language phrases are matched to temporal logic formulae (or other logics for expressing the ordering of events) can be found in Dwyer, Avrunin, and Corbett [DAC98].

(A good topic for a Paper Presentation)

Today's Agenda

- Model checking overview
- Linear temporal logic
  - Handshaking example handout
- Kripke structures
- Computation tree logic (CTL)

Next class: SMV

[Kur94] R. P. Kurshan. Computer-Aided Verification of Coordinating Processes: The Automata-Theoretic Approach. Princeton University Press, 1994.

[Pnu77] A. Pnueli. A temporal logic of programs. In 18th IEEE Symposium on Foundations of Computer Science, pages 46–57, 1977.

[QS81] J. P. Quielle and J. Sifakis. Specification and verification of concurrent systems in cesar. In Proceedings of the Fifth Internationl Symposium on Programming, volume 137 of Lecture Notes In Computer Science, pages 337–350, 1981.

[SB95] C.-J. H. Seger and R. E. Bryant. Formal verification by symbolic evaluation of partially-ordered trajectories. Formal Methods in System Design, 6:147–189, March 1995.

References

[BCM+92] J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L. J. Hwang. Symbolic model checking: 10^{20} states and beyond. Information and Computation, 98:142–170, June 1992.

[CE81] E. M. Clarke and E.A. Emerson. Synthesis of synchronization skeletons for branching time temporal logic. In Logic of Programs: Workshop, volume 131 of Lecture Notes in Computer Science. Springer-Verlag, May 1981.

[CGP99] Edmund Clarke Jr., Orna Grumberg, and Doron A. Peled. Model Checking. MIT Press, 1999.

[DAC98] Matthew B. Dwyer, George S. Avrunin, and James C. Corbett. Property specification patterns for finite-state verification. In Mark Ardis, editor, Proceedings of FMSP'98. The Second Workshop on Formal Methods in Software Practice, pages 7–15. ACM Press, March 1998.

[Hol] Gerard J. Holzmann. The Spin Model Checker: Primer and Reference Manual. Addison-Wesley.

[HR04] Michael R. A. Huth and Mark D. Ryan. Logic in Computer Science. Cambridge University Press, Cambridge, 2004. Second Edition.