CSE 620: Selective Topics

Introduction to Formal Verification

Master Studies in CSE

Fall 2017

Lecture #9

Dr. Hazem Ibrahim Shehata

Assistant Professor

Dept. of Computer & Systems Engineering

Course Outline

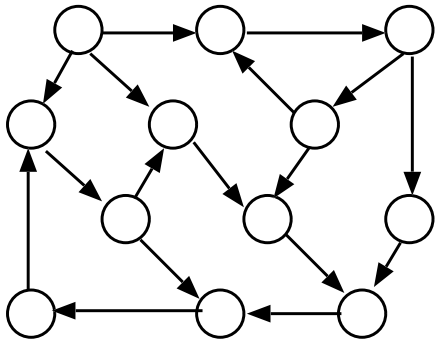
- Computational Boolean Algebra
  - —Basics
    - – Shannon Expansion
    - – Boolean Difference
    - – Quantification Operators
      - + Application to Logic Network Repair
  - —Validity Checking (Tautology Checking)
  - —Binary Decision Diagrams (BDD's)
  - —Satisfiability Checking (SAT solving)
- Model Checking
  - —Temporal Logics ➔ LTL - CTL
  - —SMV: Symbolic Model Verifier
  - —Model Checking Algorithms ➔ Explicit CTL

Model Checking

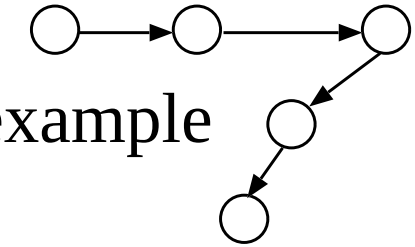
Finite Model

Kripke Structures

SMV Modelling Notation



Property Not Satisfied



Counterexample

Model Checker

G c

G (a => X b)

Temporal

Linear Temporal Logic

Property Satisfied

Algorithms: Roadmap

Problem

Given a model $\mathcal{M}$ (usually a Kripke structure) that represents the behaviour of a system, a state s, and a temporal logic formula f that represents a desired property of the system, determine whether the model and state satisfy the formula:

$$\mathcal{M} \models f$$

Model checking algorithms that we'll study:

- Explicit CTL Model Checking: labelling a graph
- Symbolic CTL Model Checking: representing sets of states using Boolean functions

Explicit CTL Model Checking

- Explicit representation of the Kripke structure as a labeled, directed graph with arcs given by pointers.
- Nodes represent the states in S
- Arcs represent the transition relation R
- Labels associated with the nodes are given by the function $L : S \rightarrow 2^{\mathrm{AP}}$
- Introduce another labelling function, *label*, to represent temporal formulae that are true of that state

Sources: Clarke, Emerson, Sistla [R10], Clarke, Grumberg, Peled [R11], Huth and Ryan [R18]

Explicit CTL M/C

Start with all states labeled with the atomic propositions that are true in those states

$$label(s) := L(s)$$

Continue to label states with the temporal formulae that we find are true of that state.

The algorithm works recursively over the structure of the formula.

Explicit CTL M/C

The depth of a formula is the number of levels of subformulae. A depth of 0 means there are no subformulae. A depth of 1 means there is one level of subformulae.

1. Decompose formula f into subformulae
2. $label(s) := L(s)$ labels all states that satisfy subformulae of depth 0
3. Find all states that satisfy subformulae of depth 1 and add the subformula to the labels of the states where it is true.
   ⋮

Explicit CTL M/C

n +1. Find all states that satisfy subformulae of depth n (which is formula f) and add the subformula to the labels of the states where it is true.

n +2. Test whether $label(s)$ contains formula f

$$\mathcal{M}, s \models f \text{ iff } f \in label(s)$$

To check whether a formula holds of all reachable states of the model, check whether the initial states are all labelled with f.

$$\mathcal{M} \models f \quad \text{iff} \quad \forall s_0 \in S_0 \bullet s_0 \in \{s \mid \mathcal{M}, s \models f\}$$

Explicit CTL M/C

Any CTL formula can be expressed in terms of $\neg, \vee, \mathbf{EX}, \mathbf{EU}, \mathbf{EG}$.

$f_1 \wedge f_2 = \neg(\neg f_1 \vee \neg f_2)$

$\mathbf{EF}\ f = \mathbf{E}[\mathbf{true}\ \mathbf{U}\ f]$

$\mathbf{AX}\ f = \neg(\mathbf{EX}\ \neg f)$
$\mathbf{AF}\ f = \neg(\mathbf{EG}\ \neg f)$
$\mathbf{AG}\ f = \neg(\mathbf{EF}\ \neg f) = \neg(\mathbf{E}[\mathbf{true}\ \mathbf{U}\ \neg f])$
$\mathbf{A}[f\ \mathbf{U}\ g] = \neg\mathbf{E}[\neg g\ \mathbf{U}\ (\neg f \wedge \neg g)] \wedge \neg\mathbf{EG}(\neg g)$

Explicit CTL M/C

Check $\neg f_1$

Add $\neg f_1$ to all $label(s)$ if $f_1 \notin label(s)$

Complexity: $O(|S|)$ (walk over the set of states)

Check $f_1 \vee f_2$

Add $f_1 \vee f_2$ to $label(s)$ if either f_1 or f_2 are in $label(s)$

Complexity: $O(|S|)$ (walk over the set of states)

Recall: S is the set of states, R is the transition relation. $|S|$ is the size of the state space.

Explicit CTL M/C: EX

Label every state that has some successor labelled by f_1.

CheckEX(f_1)

$K = \{s \mid f_1 \in \mathit{label}(s)\}$;
while $K \neq \emptyset$ do
  choose $s \in K$;
  $K := K \setminus \{s\}$;
  for all $(t, s) \in R$ do
    if $\mathbf{EX} f_1 \notin \mathit{label}(t)$ then
      $\mathit{label}(t) := \mathit{label}(t) \cup \{\mathbf{EX} f_1\}$;

Complexity: $O(|S| + |R|)$

Explicit CTL M/C: EU

Find all states that are labelled with f_2. Work backwards using R to find all states that can be reached by some path in which each state is labelled with f_1. Label all these states with $\mathbf{E}[f_1\ \mathbf{U}\ f_2]$.

Explicit CTL M/C: EU

CheckEU(f_1, f_2)

$K := \{s \mid f_2 \in label(s)\}$;
for all $s \in K$ do $label(s) := label(s) \cup \{\mathbf{E}[f_1\ \mathbf{U}\ f_2]\}$;
while $K \neq \emptyset$ do
  choose $s \in K$;
  $K := K \setminus \{s\}$;
  for all $(t, s) \in R$ do
    if $\mathbf{E}[f_1\ \mathbf{U}\ f_2] \notin label(t)$ and $f_1 \in label(t)$ then
      $label(t) := label(t) \cup \{\mathbf{E}[f_1\ \mathbf{U}\ f_2]\}$;
      $K := K \cup \{t\}$;

Complexity: $O(|S| + |R|)$

Explicit CTL M/C: EG (First Try)

EG f_1: Label all states labelled with f_1 with **EG** f_1 Repeat: delete the label **EG** f_1 from any state if none of its successors is labelled with **EG** f_1; until there is no change.

CheckEG(f_1)

$K = \{s \mid f_1 \in \mathit{label}(s)\}$;
for all $s \in K$ do $\mathit{label}(s) := \mathit{label}(s) \cup \{\mathbf{EG} f_1\}$;
do
  $K := \{s \mid \mathbf{EG} f_1 \in \mathit{label}(s)\}$;
  for all $t \in K$ do
    $\mathit{label}(t) := \mathit{label}(t) - \{\mathbf{EG} f_1\}$;
    for all $(t, s) \in R$ do
      if $s \in K$ then $\mathit{label}(t) := \mathit{label}(t) \cup \{\mathbf{EG} f_1\}$;
until $K = \{s \mid \mathbf{EG} f_1 \in \mathit{label}(s)\}$;

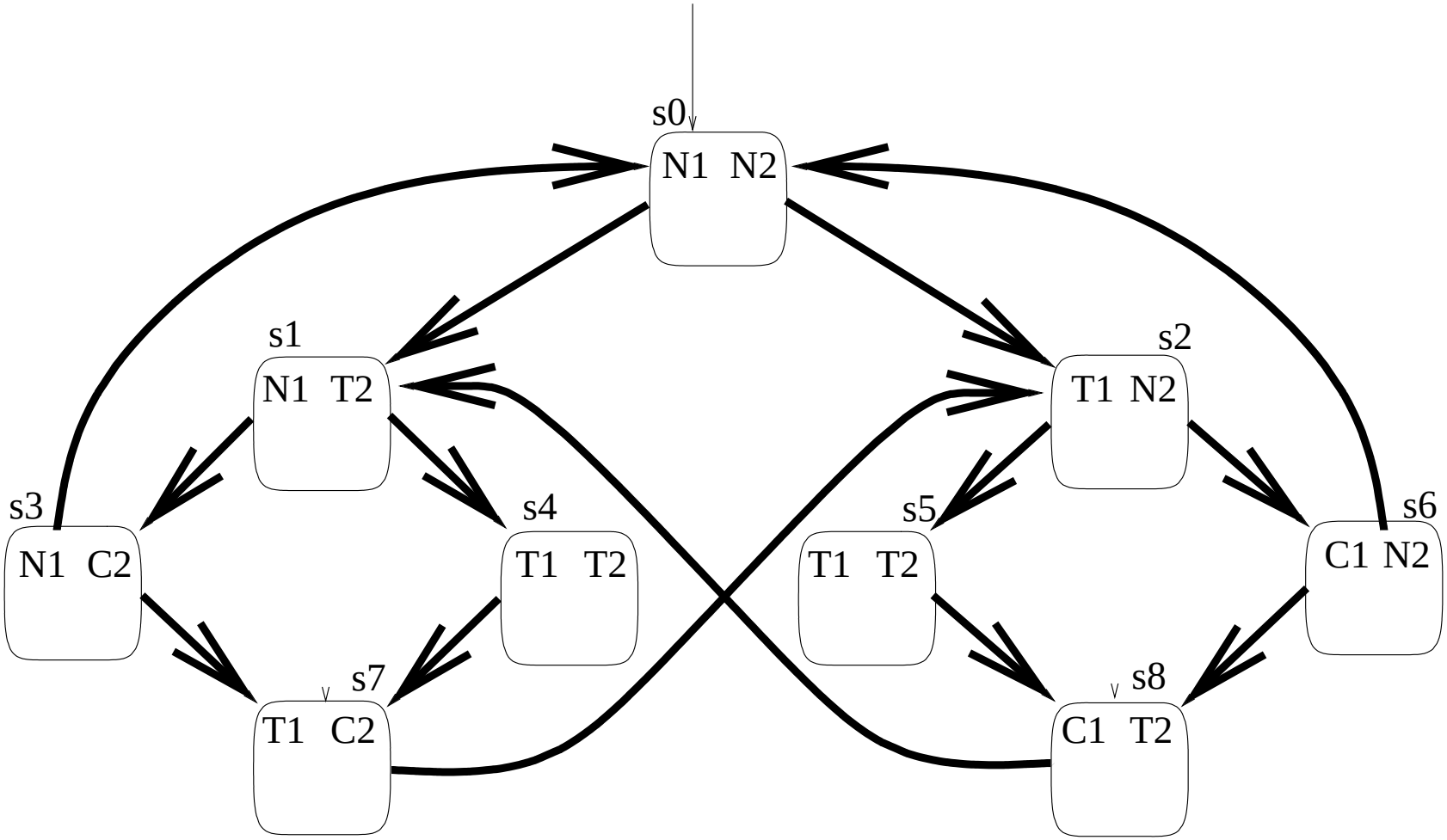
Example: Mutual Exclusion

Two processes:

- Each can be in one of three states, which they cycle through in the following order:
  1. non-critical (N)
  2. trying to enter the critical state (T)
  3. critical (C)
- interleave their steps with each other.

Both start in their non-critical sections of code.

Example: Mutual Exclusion



s_4 and s_5 are distinguished by which process was trying to get into its critical section first.

Properties of Mutual Exclusion

1. (safety) Only one process can be in its critical section at any time.

2. (liveness) Whenever a process wants to enter its critical section, it will eventually be permitted to do so.

Properties of Mutual Exclusion

1. (safety) Only one process can be in its critical section at any time.

$$\mathbf{AG}\ \neg(C1 \wedge C2)$$

alternatively:

$$\neg\mathbf{E}[\mathbf{true}\ \mathbf{U}\ \neg(\neg C1 \vee \neg C2)]$$

2. (liveness) Whenever a process wants to enter its critical section, it will eventually be permitted to do so.

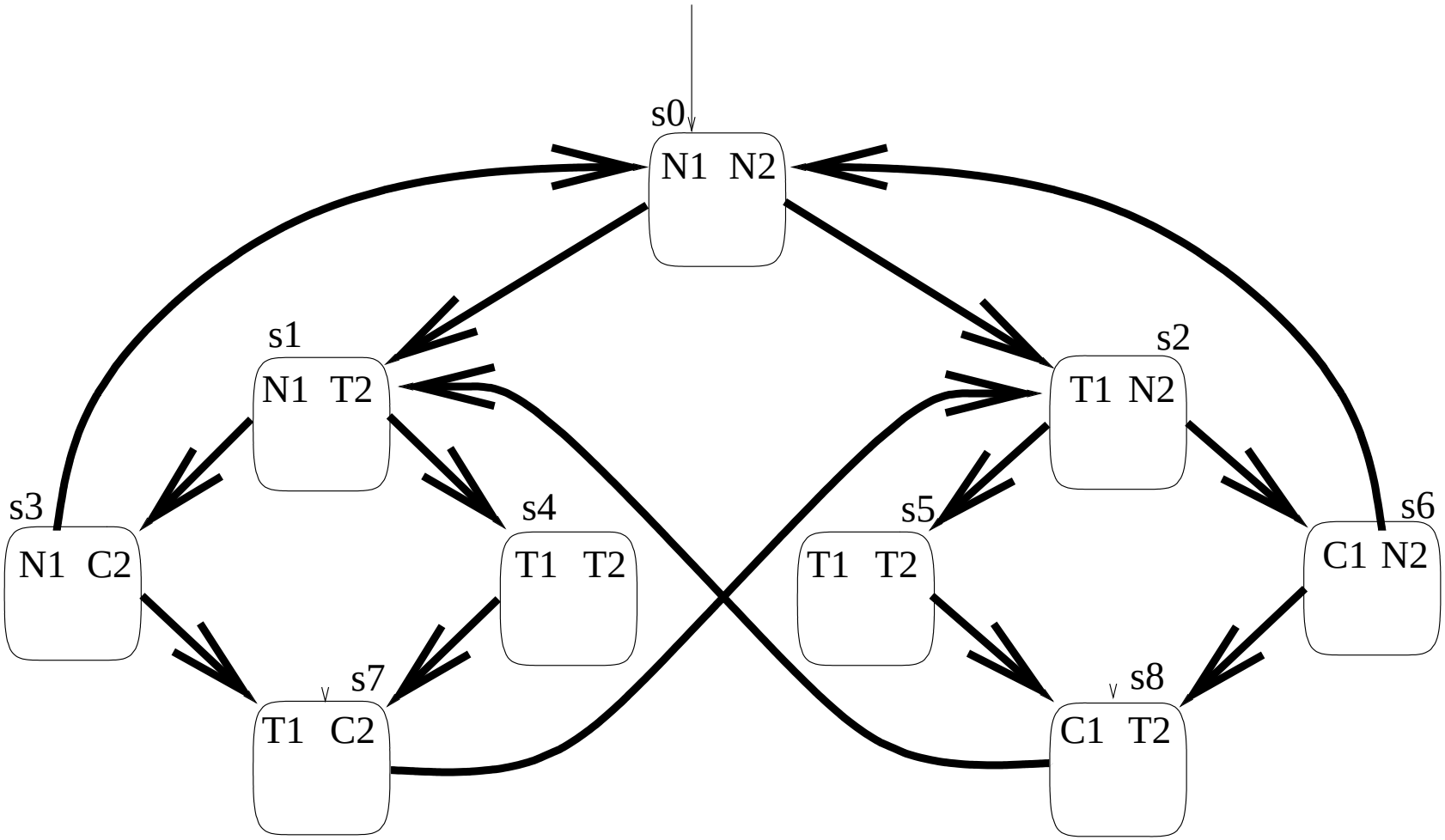
$$\mathbf{AG}(T1 \Rightarrow \mathbf{AF}\ C1)$$

alternatively:

$$\neg(\mathbf{E}[\mathbf{true}\ \mathbf{U}\ \neg(\neg T1 \vee \neg(\mathbf{EG}\ \neg C1))])$$

Checking Safety

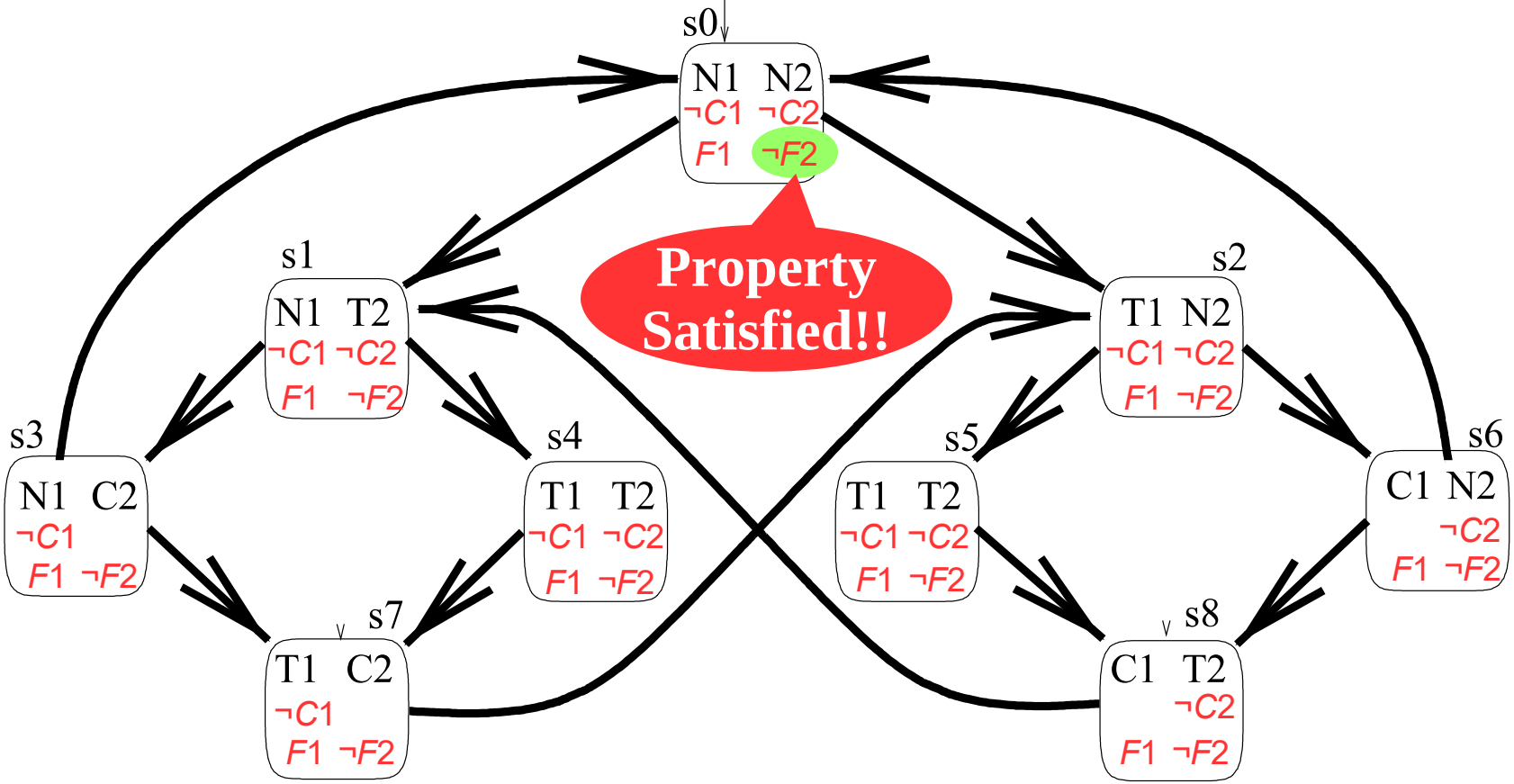
$$\neg \mathbf{E}[\mathbf{true}\ \mathbf{U}\ \neg(\neg C1 \vee \neg C2)]$$



Checking Safety

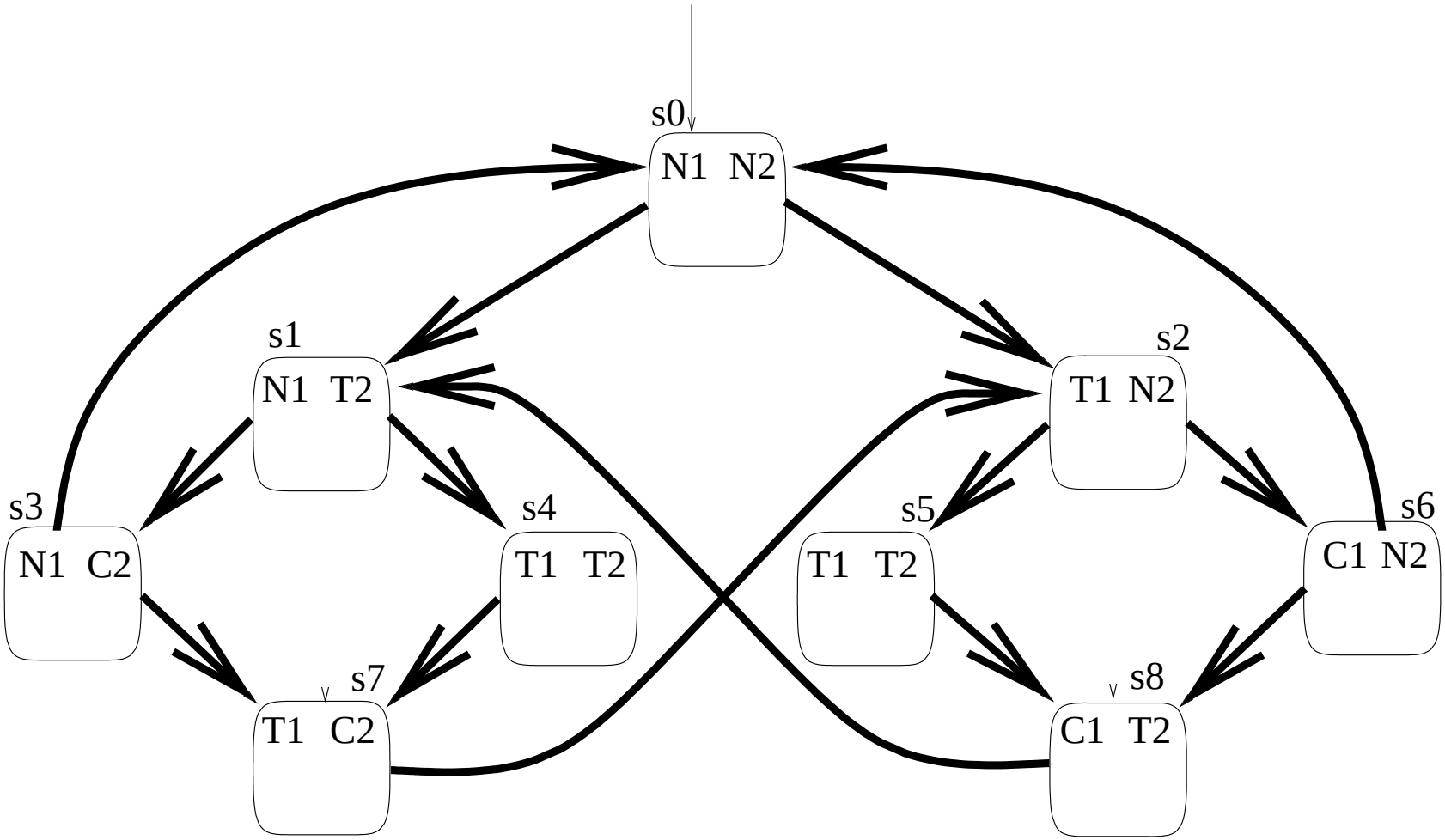
$$\neg \mathbf{E}[\mathbf{true}\ \mathbf{U}\ \underbrace{\neg(\neg C1 \vee \neg C2)}_{F1}]$$

$F2$

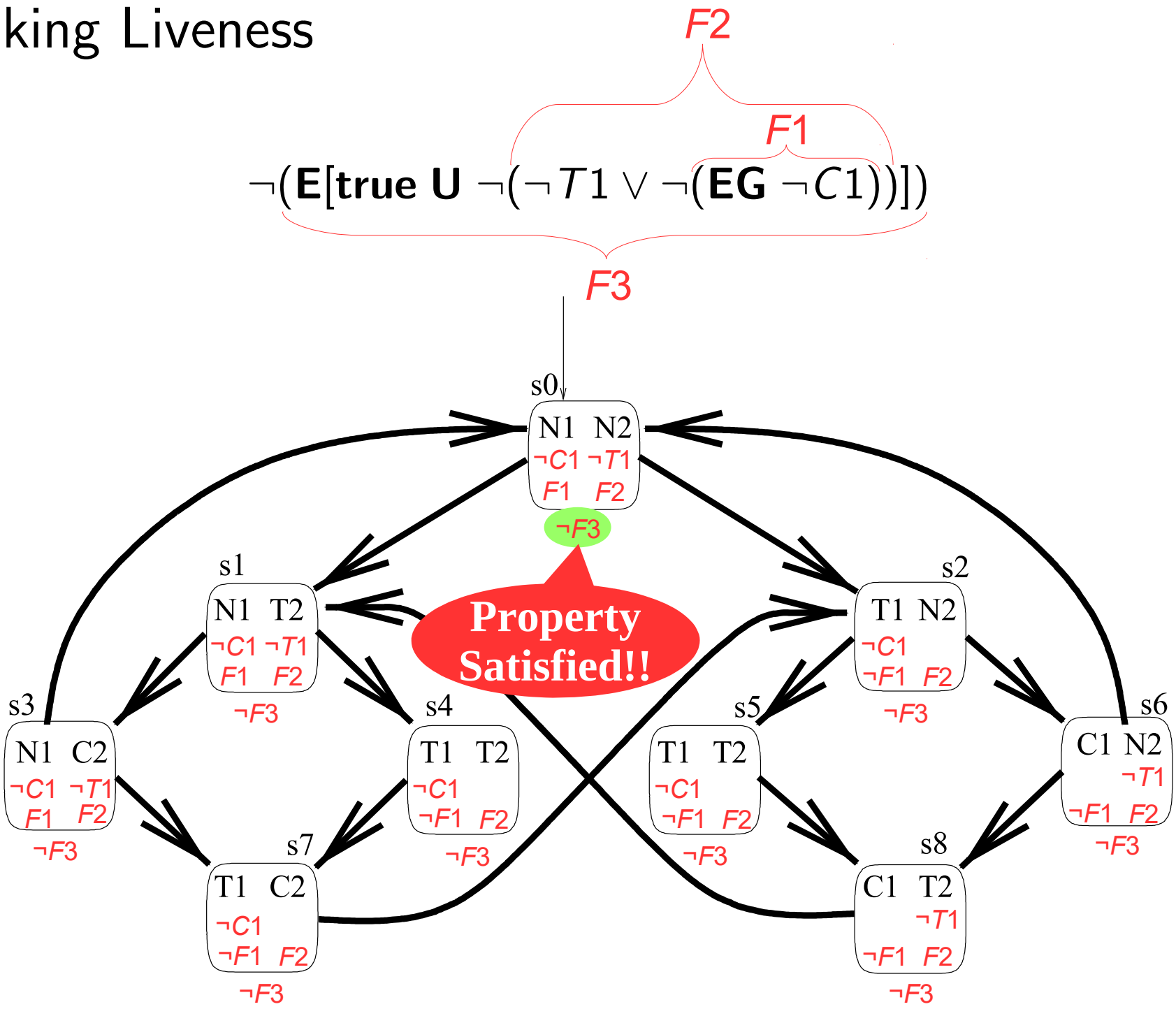


Checking Liveness

$$\neg(\mathbf{E}[\mathbf{true}\ \mathbf{U}\ \neg(\neg T1 \vee \neg(\mathbf{EG}\ \neg C1))])$$



Checking Liveness



Algorithm Complexity

- Worst case complexity of checking each subformula: $O(|S| + |R|)$
- For a formula of length n, there are n sub-formulae to check.
- Worst case complexity to check a formula of length n is $O(n \times (|S| + |R|))$

Thus, complexity is linear in the size of the formula and in the size of the state space. But the size of the state space is exponential in the number of parallel system components and in the number of variables!

State Space Explosion Problem