HARVARD

School of Engineering and Applied Sciences

Model checking

CS252r Spring 2011

Contains material from slides by Edmund Clarke (http://www.cs.cmu.edu/~emc/15817-s05/)

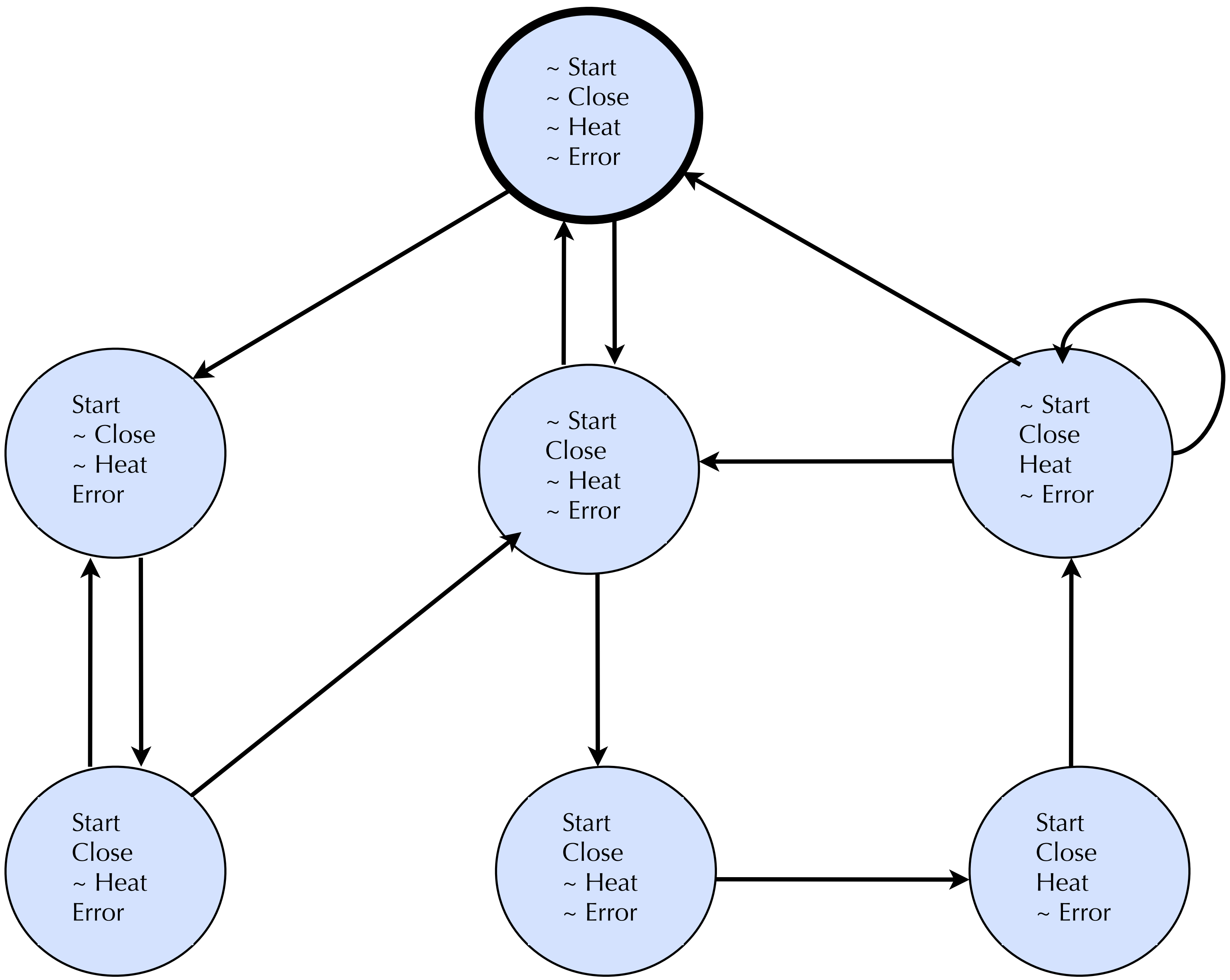
What is model checking?

- Automatic verification technique for finite state systems
  - Specifications for the system are written in **temporal logic**
  - Exhaustively search state space, to ensure that specification is satisfied
- Typically applied to hardware designs
  - Temporal logic can express safety requirements for concurrent systems
- In last 10-15 years, interest in applying to software
- Developed in 1980's by Clarke, Emerson, and Sistla and by Queille and Sifakis
  - Clarke, Emerson, and Sifakis got Turing Award in 2007

Example

- From Edmund Clarke http://www.cs.cmu.edu/~emc/15817-s05/
- Microwave oven states
  - Four atomic propositions
    - Start: “start” button pressed
    - Close: is door closed?
    - Heat: Microwave active
    - Error: error state

Example


~ Start
~ Close
~ Heat
~ Error
Start
~ Close
~ Heat
Error
~ Start
Close
~ Heat
~ Error
~ Start
Close
Heat
~ Error
Start
Close
~ Heat
Error
Start
Close
~ Heat
~ Error
Start
Close
Heat
~ Error

Example

- Temporal safety property: the oven doesn't heat up until the door is closed
  - Not heat holds until door is closed
  - $(\neg \text{Heat}) \ U \ \text{Close}$

Temporal logic

- A kind of **modal** logic
  - Modal logics originally developed to express modalities such as necessity and possibility
  - Also used to reason about
    - knowledge (with much success in reasoning about distributed systems, distributed protocols, and security)
    - permission and obligation
    - ...
- Temporal logics reason about what is true, when
  - Each atomic proposition is either true or false in a given state
  - Consider the execution of a system as a sequence of states
  - The "current time" is an index into the sequence
    - The future is later indices, the past is earlier indices

Temporal logic syntax

- p: Primitive propositions
- Standard Boolean connectives ($\neg$, $\vee$, $\wedge$, $\Rightarrow$)
- Temporal operators
  - Gφ — φ is always true (i.e., now, and in the future); Globally
  - Fφ — φ is true sometime in the Future
  - Xφ — φ is true in the neXt time step
  - φUψ — φ is true Until ψ is true

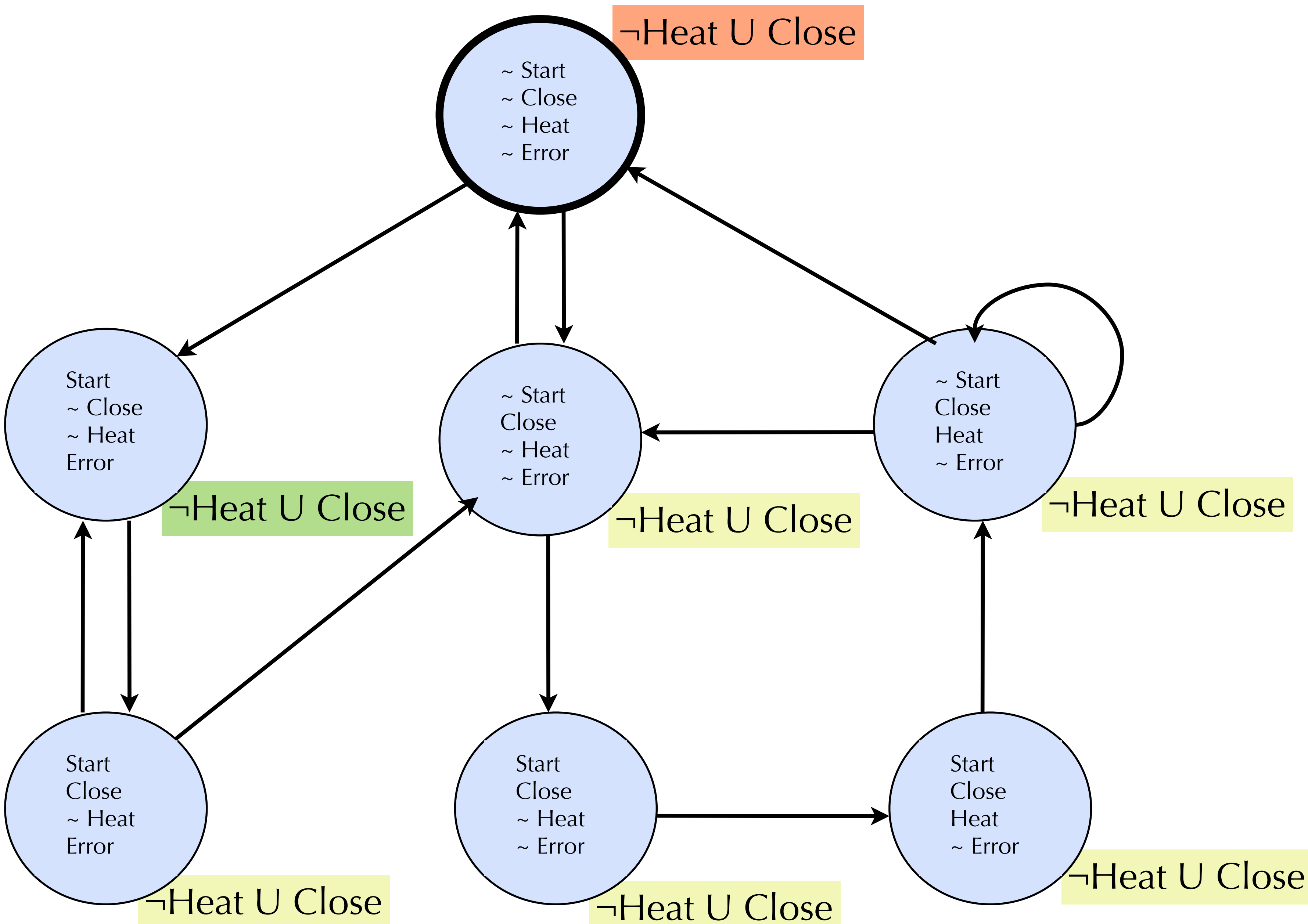
LTL semantics

- $x, i \vDash \varphi$ — Given execution sequence x, at time i, φ is true

- $x, i \vDash p$ — iff p is true in state x_i
- $x, i \vDash \varphi \land \psi$ — iff $x, i \vDash \varphi$ and $x, i \vDash \psi$
- $x, i \vDash X\varphi$ — iff $x, i+1 \vDash \varphi$
- $x, i \vDash G\varphi$ — iff $x, j \vDash \varphi$ for all $j \geq i$
- $x, i \vDash F\varphi$ — iff $x, j \vDash \varphi$ for some $j \geq i$
- $x, i \vDash \varphi U \psi$ — iff $x, j \vDash \psi$ for some $j \geq i$, and for all $j > k \geq i$ we have $x, k \vDash \varphi$

Example

¬Heat U Close



Model Checking Problem

- Given state transition graph M
- Let φ be specification (a temporal logic formula)
- Find all states s of M such that for all execution sequences x starting from s, $x,0 \vDash \varphi$

- Efficient algorithms for solving this
  - (Num states + Num transitions) × $2^{O(|\varphi|)}$

State explosion

- Big problem: state explosion
- Consider concurrent system
  - Exponentially many different states arise due to exponentially many possible interleavings
- Many software systems have infinite state space...

Addressing state explosion

- Symbolic model checking
  - Used by all "real" model checkers
  - Use boolean encoding of state space
    - Allows for efficient representation of states and transitions through BDDs
    - Scales up to hundreds of state variables
      - Systems with 10^{120} reachable states have been checked
- But what about software with infinite state space? with non-boolean-valued data?
- Abstraction
  - Use a finite abstraction of the software
  - See next class for a method of automatically discovering an appropriate abstraction

Addressing state explosion

- Other techniques
  - Bounded model checking
  - Compositional reasoning
  - Symmetry
  - “Cone of influence”
  - …

Tools

- Early tools
  - EMC (Clarke, Emerson, and Sistla)
  - Caesar (Queille, Sifakis)
- Many modern tools, for many languages
  - SPIN
  - SLAM
  - CHESS
  - BLAST
  - ...

More on temporal logic

- Logic we saw previously is known as **Linear temporal logic** (LTL)
  - Semantics defined over a single execution trace
- Another popular temporal logic is **Computation tree logic** (CTL) aka Branching time logic
  - Semantics defined over a tree
    - i.e., may be many possible futures

CTL syntax

- p: Primitive propositions
- Standard Boolean connectives ($\neg$, $\vee$, $\wedge$, $\Rightarrow$)
- Temporal operators
  - AGφ on all paths from here, φ is always true
  - EGφ on some path from here, φ is always true
  - AFφ on all paths from here φ is true sometime in the future
  - EFφ on some path from here φ is true sometime in the future
  - AXφ on all paths from here, φ is true in the neXt time step
  - EXφ φ is true in the neXt time step
  - A[φUψ] on all paths from here, φ is true Until ψ is true
  - E[φUψ] on some path from here φ is true Until ψ is true

CTL and LTL

- Which is more expressive?
- Incomparable!
- E.g., formula FGφ cannot be expressed in CTL
  - “At some time in the future, φ is true from that time onwards)”
- E.g., AG(EFφ) cannot be expressed in LTL
  - “for all paths, it is always the case that there is some path on which φ is eventually true”

CTL*

- CTL* is strictly more powerful than both CTL and LTL
- Allows temporal operators (X, F, G, U) to be used without path quantifiers (A and E)
- Syntax
  - $\varphi ::= p \mid \varphi_1 \vee \varphi_2 \mid \ldots \mid A\psi \mid E\psi$
  - $\psi ::= \varphi \mid \psi_1 \vee \psi_2 \mid \ldots \mid G\psi \mid F\psi \mid X\psi \mid \psi_1 U \psi_2$
- Semantics
  - $(M, s) \vDash A\psi$ iff for all paths x from s in M $x,0 \vDash \psi$
  - ...
  - $x,i \vDash \varphi$ iff $(M,x_i) \vDash \psi$
  - $x,i \vDash X\varphi$ iff $x,i+1 \vDash \varphi$
  - ...

Modal μ-calculus

- A very powerful logic, adds fix-point operators
- A form of **dynamic logic**
  - Reasons about how actions a affect state
- Syntax
  - $\varphi ::= p \mid \varphi1 \vee \varphi2 \mid \ldots \mid [a]\varphi \mid \langle a \rangle\varphi \mid X \mid \mu X.\ \varphi(X) \mid \nu X.\ \varphi(X)$
  - $[a]\varphi$ means for all states reachable by performing a single a action, φ is true
  - $[K]\varphi$ means for all states reachable by performing any single action $a \in K$, φ is true
  - $\langle a \rangle\varphi$ means for some state reachable by performing a single a action, φ is true
  - (In CTL, $EX\varphi$ and $AX\varphi$ is like $\langle L \rangle\varphi$ and $[L]\varphi$, respectively, where L is set of all actions)
  - $\mu X.\ \varphi(X)$ is **least fixed point** of φ, $\nu X.\ \varphi(X)$ is **greatest fixed point** of φ,

Encoding CTL in modal μ

- All operators in CTL (AX, AF, AG, A[U], EX, EF, EG, E [U]) can be encoded using A[U], E[U], EX
  - $AF\varphi \equiv A[\text{true}\ U\ \varphi]$
  - $EF\varphi \equiv E[\text{true}\ U\ \varphi]$
  - $AG\varphi \equiv \neg E(\text{true}\ U\ \neg\varphi)$
  - $EG\varphi \equiv \neg A(\text{true}\ U\ \neg\varphi)$
  - $AX\varphi \equiv \neg EX\neg\varphi$
- Encoding in modal μ
  - $[\![EX\varphi]\!] = \langle L\rangle [\![\varphi]\!]$
  - $[\![\ A[\varphi\ U\ \psi]\ ]\!] = \mu X.\ [\![\psi]\!] \vee ([\![\varphi]\!] \wedge [L]X)$
  - $[\![\ E[\varphi\ U\ \psi]\ ]\!] = \mu X.\ [\![\psi]\!] \vee ([\![\varphi]\!] \wedge \langle L\rangle X)$

Summary of temporal logics