Model Checking

Nick Sumner
wsumner@sfu.ca
Technique limitations vs risks

- Most techniques have limited claims
Technique limitations vs risks

- Most techniques have limited claims
  - Testing
    - Show the program can behave correctly
    - Provide confidence for given criteria
Technique limitations vs risks

• Most techniques have limited claims
  – Testing
    • Show the program can behave correctly
    • Provide confidence for given criteria
  – Dynamic Analysis
    • Helps find bugs
    • Collect information about programs
Technique limitations vs risks

- Most techniques have limited claims
  - Testing
    - Show the program can behave correctly
    - Provide confidence for given criteria
  - Dynamic Analysis
    - Helps find bugs
    - Collect information about programs
  - Code Review
    - Finds many surface level issues
Technique limitations vs risks

- Most techniques have limited claims
  - **Testing**
    - Show the program can behave correctly
    - Provide confidence for given criteria
  - **Dynamic Analysis**
    - Helps find bugs
    - Collect information about programs
  - **Code Review**
    - Finds many surface level issues

- Programs may exhibit subtle, hard to identify issues
  - Distributed file system integrity
  - Coordinating telephony
Technique limitations vs risks

- Most techniques have limited claims
  - Testing
    - Show the program can behave correctly
    - Provide confidence for given criteria
  - Dynamic Analysis
    - Helps find bugs
    - Collect information about programs
  - Code Review
    - Finds many surface level issues

- Programs may exhibit subtle, hard to identify issues
  - Distributed file system integrity
  - Coordinating telephony

- But what if these components are mission critical?!
Proving properties of programs

- If a particular property is mission critical, it may be worth *proving* that the property holds.
If a particular property is mission critical, it may be worth proving that the property holds. This often has a higher cost than applying other tools.
If a particular property is mission critical, it may be worth proving that the property holds.

- This often has a higher cost than applying other tools.
- The ROI for mission critical infrastructure can pay off.
Proving properties of programs

• If a particular property is mission critical, it may be worth proving that the property holds
  – This often has a higher cost than applying other tools
  – The ROI for mission critical infrastructure can pay off

• What kinds of properties would be interesting?
Proving properties of programs

- If a particular property is mission critical, it may be worth *proving* that the property holds
  - This often has a higher cost than applying other tools
  - The ROI for mission critical infrastructure can pay off

- What kinds of properties would be interesting?
  - *Safety* – Something bad never happens
Proving properties of programs

- If a particular property is mission critical, it may be worth proving that the property holds
  - This often has a higher cost than applying other tools
  - The ROI for mission critical infrastructure can pay off

- What kinds of properties would be interesting?
  - **Safety** – Something bad never happens
  - **Liveness** – Something good eventually happens
Proving properties of programs

- If a particular property is mission critical, it may be worth proving that the property holds
  - This often has a higher cost than applying other tools
  - The ROI for mission critical infrastructure can pay off

- What kinds of properties would be interesting?
  - **Safety** – Something bad never happens
  - **Liveness** – Something good eventually happens

  Some things should always be true (invariants), while others should eventually be true.
Proving properties of programs

- If a particular property is mission critical, it may be worth *proving* that the property holds
  - This often has a higher cost than applying other tools
  - The ROI for mission critical infrastructure can pay off

- What kinds of properties would be interesting?
  - *Safety* – Something bad never happens
  - *Liveness* – Something good eventually happens

- *Some things should always be true* (invariants), while others should eventually be true.

- *Model checking* is one such tool for proving these properties
Model Checking Overview

- Model checking is an automated technique for proving properties of finite state systems.
Model Checking Overview

- Model checking is an automated technique for proving properties of finite state systems.
Model Checking Overview

- Model checking is an automated technique for proving properties of finite state systems.
Model checking is an automated technique for proving properties of finite state systems.
Model checking is an automated technique for proving properties of finite state systems.
Model checking is an automated technique for proving properties of finite state systems.

Diagram:
- **System Design** → **Implementation**
- **Formal Property Specification** → **Formal Abstract Model**
- **Verifier** → **Proof**
Model Checking Overview

- Model checking is an automated technique for proving properties of finite state systems
Model Checking Overview

- Model checking is an automated technique for proving properties of finite state systems.
How can we specify the model?

- Often written in a formal specification language
  - temporal logic (CTL, LTL), Alloy, TLA, ...
How can we specify the model?

- Often written in a formal specification language
  - temporal logic (CTL, LTL), Alloy, TLA, ...
- Need to express the finite states & transitions
How can we specify the model?

• Often written in a formal specification language
  – temporal logic (CTL, LTL), Alloy, TLA, ...

• Need to express the finite states & transitions
  – (Oven example from Edmund Clarke)

Start
Close
Heat
Error
How can we specify the model?

- Often written in a formal specification language
  - temporal logic (CTL, LTL), Alloy, TLA, ...
- Need to express the finite states & transitions
  - (Oven example from Edmund Clarke)
How can we specify the model?

- Often written in a formal specification language
  - temporal logic (CTL, LTL), Alloy, TLA, ...
- Need to express the finite states & transitions
  - (Oven example from Edmund Clarke)
How can we specify the model?

- Often written in a formal specification language
  - temporal logic (CTL, LTL), Alloy, TLA, ...

- Need to express the finite states & transitions
  - (Oven example from Edmund Clarke)
How can we specify the model?

- Often written in a formal specification language
  - temporal logic (CTL, LTL), Alloy, TLA, ...
- Need to express the finite states & transitions
  - (Oven example from Edmund Clarke)
How can we specify the model?

- Often written in a formal specification language
  - temporal logic (CTL, LTL), Alloy, TLA, ...
- Need to express the finite states & transitions
  - (Oven example from Edmund Clarke)
How can we specify the model?

- Often written in a formal specification language
  - temporal logic (CTL, LTL), Alloy, TLA, ...
- Need to express the finite states & transitions
  - (Oven example from Edmund Clarke)
How can we specify the model?

- Often written in a formal specification language
  - temporal logic (CTL, LTL), Alloy, TLA, ...
- Need to express the finite states & transitions
  - (Oven example from Edmund Clarke)
How can we specify the model?

- Often written in a formal specification language
  - temporal logic (CTL, LTL), Alloy, TLA, ...
- Need to express the finite states & transitions
  - (Oven example from Edmund Clarke)
How can we specify the model?

- Often written in a formal specification language
  - temporal logic (CTL, LTL), Alloy, TLA, ...
- Need to express the finite states & transitions
  - (Oven example from Edmund Clarke)
How can we specify properties?

- Often in the same language, inspired by temporal logic
How can we specify properties?

- Often in the same language, inspired by temporal logic
- Temporal constraints help express properties particularly interesting to concurrent and distributed systems
How can we specify properties?

- Often in the same language, inspired by temporal logic

- Temporal constraints help express properties particularly interesting to concurrent and distributed systems
  - e.g. The oven doesn’t heat up until the door is closed
How can we specify properties?

- Often in the same language, inspired by temporal logic
- Temporal constraints help express properties particularly interesting to concurrent and distributed systems
  - e.g. The oven doesn’t heat up until the door is closed
- Temporal constraints for a proposition p:
How can we specify properties?

- Often in the same language, inspired by temporal logic

- Temporal constraints help express properties particularly interesting to concurrent and distributed systems
  - e.g. The oven doesn’t heat up until the door is closed

- Temporal constraints for a proposition $p$:
  - $p$ will hold eventually in the future
How can we specify properties?

- Often in the same language, inspired by temporal logic
- Temporal constraints help express properties particularly interesting to concurrent and distributed systems
  - e.g. The oven doesn’t heat up until the door is closed
- Temporal constraints for a proposition $p$:
  - $p$ will hold eventually in the future
  - $p$ holds in all future states
How can we specify properties?

- Often in the same language, inspired by temporal logic

- Temporal constraints help express properties particularly interesting to concurrent and distributed systems
  - e.g. The oven doesn’t heat up until the door is closed

- Temporal constraints for a proposition $p$:
  - $p$ will hold eventually in the future
  - $p$ holds in all future states
  - $p$ holds in the next state
How can we specify properties?

- Often in the same language, inspired by temporal logic

- Temporal constraints help express properties particularly interesting to concurrent and distributed systems
  - e.g. The oven doesn’t heat up until the door is closed

- Temporal constraints for a proposition p:
  - p will hold eventually in the future
  - p holds in all future states
  - p holds in the next state
  - p holds until another proposition q holds
Traffic Lights

- Traffic lights are a common application of safety critical embedded systems
Traffic Lights

- Traffic lights are a common application of safety critical embedded systems
Traffic Lights

- Traffic lights are a common application of safety critical embedded systems
- Interesting properties?
Traffic Lights

- Traffic lights are a common application of safety critical embedded systems
- Interesting properties
  - The light is green infinitely often
Traffic Lights

- Traffic lights are a common application of safety critical embedded systems

- Interesting properties
  - The light is green infinitely often
  - A red light does not immediately become green
  - ...

![Traffic Light States Diagram]
Traffic Lights

- Traffic lights are a common application of safety critical embedded systems
- Interesting properties
  - The light is green infinitely often
  - A red light does not immediately become green
  - ...
- You can also specify lights at an intersection as a distributed system & check the consistency!
Do people actually use it?

- Aerospace
- Hardware
- Critical infrastructure providers (including Amazon)
- Microsoft holds internal (& external) lectures on it
Do people actually use it?

- Aerospace
- Hardware
- Critical infrastructure providers (including Amazon)
- Microsoft holds internal (& external) lectures on it

Amazon’s experience (Using TLA+)

- Now used by several teams within AWS
- Each system has a 1-2KLOC TLA+ specification
- Detected several internal issues before they struck
Do people actually use it?

- Aerospace
- Hardware
- Critical infrastructure providers (including Amazon)
- Microsoft holds internal (& external) lectures on it

Amazon’s experience (Using TLA+)
- Now used by several teams within AWS
- Each system has a 1-2KLOC TLA+ specification
- Detected several internal issues before they struck

It is increasingly desirable for platform providers
What does TLA+ look like?

- Let’s walk through an example...
Summary

- Model checking can be an excellent way of proving properties about programs.
Summary

- Model checking can be an excellent way of proving properties about programs.
- While it requires more effort and cost, it can prevent critical issues.
Model checking can be an excellent way of proving properties about programs.

While it requires more effort and cost, it can prevent critical issues.

One such platform for model checking is TLA+.