Systeme Hoher Qualität und Sicherheit
Vorlesung 11 vom 13.01.2014: Modelchecking with LTL and CTL

Christoph Lüth & Christian Liguda

Universität Bremen

Wintersemester 2013/14

Organisatorisches

- Noch ein Übungsblatt?
- Prüfungen — KW 06 (4./5. Feb.)

Where are we?

Introduction

- Last lectures: verifying program properties with the **Floyd-Hoare** calculus
- In the Floyd-Hoare calculus, program verification is reduced to a **deductive** problem by translating the program into logic (specifically, state change becomes substitution).
- Model-checking takes a different approach: the system is modelled directly by a finite-state machine, and properties are expressed in some logic for FSM. Program verification reduces to state enumeration, which can be done automatically.
- The logics we will considere here are temporal logic: linear temporal logic (**LTL**) and branching temporal logic (**CTL**)

The Model-Checking Problem

The Basic Question

Given a model $\mathcal{M}$, and a property ϕ, we want to know whether

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

- What is $\mathcal{M}$? **Finite state machines**
- What is ϕ? **Temporal logic**
- How to prove it? Enumerating states — **model checking**

Finite State Machines

Finite State Machine (FSM)

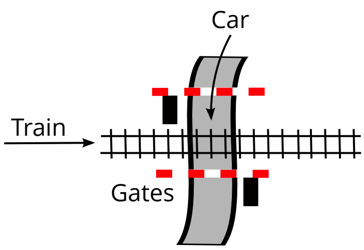
A FSM is given by $\mathcal{M} = \langle \Sigma, \rightarrow \rangle$ where

- Σ is a finite set of **states**, and
- $\rightarrow \subseteq \Sigma \times \Sigma$ is a **transition relation**, such that $\rightarrow$ is left-total:

$$\forall s \in \Sigma.\, \exists s' \in \Sigma.\, s \rightarrow s'$$

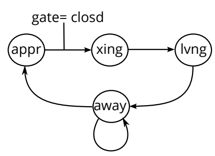
- Many variations of this definition exists, e.g. sometimes we have state variables or labelled transitions.
- Note there is no **final** state, and no input or output (this is the key difference to automata).
- If $\rightarrow$ is a function, the FSM is **deterministic**, otherwise it is **non-deterministic**.

The Railway Crossing

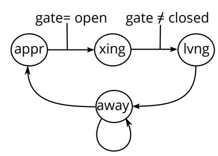


Modelling the Railway Crossing

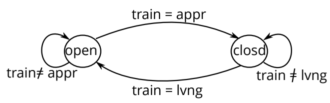
States of the train:



States of the car:



States of the gate:



The FSM

- The states here are a map from variables *Car*, *Train*, *Gate* to the domains

$$\begin{aligned}\Sigma_{Car} &= \{appr, xing, lvng, away\}\\ \Sigma_{Train} &= \{appr, xing, lvng, away\}\\ \Sigma_{Gate} &= \{open, clsd\}\end{aligned}$$

or alternatively, a three-tuple $S \in \Sigma = \Sigma_{Car} \times \Sigma_{Train} \times \Sigma_{Gate}$.

- The transition relation is given by e.g.

$$\begin{aligned}&\langle away, open, away\rangle \rightarrow \langle appr, open, away\rangle\\ &\langle appr, open, away\rangle \rightarrow \langle xing, open, away\rangle\\ &\ldots\end{aligned}$$

Railway Crossing — Safety Properties

- Now we want to express safety (or security) **properties**, such as the following:
  - Cars and trains never cross at the same time.
  - The car can always leave the crossing
  - Approaching trains may eventually cross.
  - There are cars crossing the tracks.
- We distinguish **safety** properties from **liveness** properties:
  - Safety: something bad never happens.
  - Liveness: something good will (eventually) happen.
- To express these properties, we need to talk about sequences of states in an FSM.

Linear Temporal Logic (LTL) and Paths

- LTL allows us to talk about **paths** in a FSM, where a path is a sequence of states connected by the transition relation.
- We first define the syntax of formula,
- then what it means for a path to satisfy the formula, and
- from that we derive the notion of a model for an LTL formula.

Paths

Given a FSM $\mathcal{M} = \langle \Sigma, \rightarrow \rangle$, a **path** in $\mathcal{M}$ is an (infinite) sequence $\langle s_1, s_2, s_3, \ldots \rangle$ such that $s_i \in \Sigma$ and $s_i \rightarrow s_{i+1}$ for all i.

- For a path $p = \langle s_1, s_2, s_3, \ldots \rangle$, we write p_i for s_i (selection) and p^i for $\langle s_i, s_{i+1}, \ldots \rangle$ (the suffix starting at i).

Linear Temporal Logic (LTL)

$$\begin{array}{rll} \phi ::= & \top \mid \bot \mid p & \text{— True, false, atomic}\\ \mid & \neg\phi \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid \phi_1 \longrightarrow \phi_2 & \text{— Propositional formulae}\\ \mid & \mathsf{X}\,\phi & \text{— Next state}\\ \mid & \mathsf{F}\,\phi & \text{— Some Future State}\\ \mid & \mathsf{G}\,\phi & \text{— All future states (Globally)}\\ \mid & \phi_1\ U\ \phi_2 & \text{— Until} \end{array}$$

- Operator precedence: Unary operators; then U; then $\wedge$, $\vee$; then $\longrightarrow$.
- An atomic formula p above denotes a **state predicate**. Note that different FSMs have different states, so the notion of whether an atomic formula is satisfied depends on the FSM in question. A different (but equivalent) approach is to label states with atomic propositions.
- From these, we can define other operators, such as $\phi\ R\ \psi$ (release) or $\phi\ W\ \psi$ (weak until).

Satifsaction and Models of LTL

Given a path p and an LTL formula ϕ, the **satisfaction relation** $p \models \phi$ is defined inductively as follows:

$$\begin{array}{lllllll} p & \models & True & \quad p & \models & \phi \wedge \psi \text{ iff } p \models \phi \text{ and } p \models \psi\\ p & \not\models & False & \quad p & \models & \phi \vee \psi \text{ iff } p \models \phi \text{ or } p \models \psi\\ p & \models & p \text{ iff } p(p_1) & \quad p & \models & \phi \longrightarrow \psi \text{ iff whenever } p \models \phi \text{ then } p \models \psi\\ p & \models & \neg\phi \text{ iff } p \not\models \phi & & & \end{array}$$

$$\begin{array}{lll} p & \models & \mathsf{X}\,\phi \text{ iff } p^2 \models \phi\\ p & \models & \mathsf{G}\,\phi \text{ iff for all } i\text{, we have } p^i \models \phi\\ p & \models & \mathsf{F}\,\phi \text{ iff there is } i \text{ such that } p^i \models \phi\\ p & \models & \phi\ U\ \psi \text{ iff there is } i\ p^i \models \psi \text{ and for all } j = 1, \ldots, i-1,\ \ p^j \models \phi \end{array}$$

Models of LTL formulae

A FSM $\mathcal{M}$ satisfies an LTL formula ϕ, $\mathcal{M} \models \phi$, iff every path p in $\mathcal{M}$ satisfies ϕ.

The Railway Crossing

- Cars and trains never cross at the same time.

$$\mathsf{G}\,\neg(car = xing \wedge train = xing)$$

- A car can always leave the crossing:

$$\mathsf{G}(car = xing \longrightarrow \mathsf{F}(car = lvng))$$

- Approaching trains may eventually cross:

$$\mathsf{G}(train = appr \longrightarrow \mathsf{F}(train = xing))$$

- There are cars crossing the tracks:

$\mathsf{F}(car = xing)$ means **something else**!

  - Can not express this in LTL!

Computational Tree Logic (CTL)

- LTL does not allow us the quantify over paths, e.g. assert the existance of a path satisfying a particular property.
- To a limited degree, we can solve this problem by negation: instead of asserting a property ϕ, we check wether $\neg\phi$ is satisfied; if that is not the case, ϕ holds. But this does not work for mixtures of universal and existential quantifiers.
- Computational Tree Logic (CTL) is an extension of LTL which allows this by adding universal and existential quantifiers to the modal operators.
- The name comes from considering paths in the **computational tree** obtained by **unwinding** the FSM.

CTL Formulae

$$\begin{array}{rll} \phi ::= & \top \mid \bot \mid p & \text{— True, false, atomic}\\ \mid & \neg\phi \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid \phi_1 \longrightarrow \phi_2 & \text{— Propositional formulae}\\ \mid & \mathsf{AX}\,\phi \mid \mathsf{EX}\,\phi & \text{— All or some next state}\\ \mid & \mathsf{AF}\,\phi \mid \mathsf{EF}\,\phi & \text{— All or some future states}\\ \mid & \mathsf{AG}\,\phi \mid \mathsf{EG}\,\phi & \text{— All or some global future}\\ \mid & \mathsf{A}[\phi_1\ U\ \phi_2] \mid \mathsf{E}[\phi_1\ U\ \phi_2] & \text{— Until all or some} \end{array}$$

Satisfaction

- Note that CTL formulae can be considered to be a LTL formulae with a 'modality' (A or E) added on top of each temporal operator.
- Generally speaking, the A modality says the temporal operator holds for all paths, and the E modality says the temporal operator only holds for all least one path.
  - Of course, that strictly speaking is not true, because the arguments of the temporal operators are in turn CTL forumulae, so we need recursion.
- This all explains why we do not define a satisfaction for a single path p, but satisfaction with respect to a specific **state** in an FSM.

Satisfaction for CTL

Given an FSM $\mathcal{M} = \langle \Sigma, \rightarrow \rangle$, $s \in \Sigma$ and a CTL formula ϕ, then $\mathcal{M}, s \models \phi$ is defined inductively as follows:

$$
\begin{array}{lcl}
\mathcal{M}, s & \models & \mathit{True} \\
\mathcal{M}, s & \not\models & \mathit{False} \\
\mathcal{M}, s & \models & p \text{ iff } p(s) \\
\mathcal{M}, s & \models & \phi \wedge \psi \text{ iff } \mathcal{M}, s \models \phi \text{ and } \mathcal{M}, s \models \psi \\
\mathcal{M}, s & \models & \phi \vee \psi \text{ iff } \mathcal{M}, s \models \phi \text{ or } \mathcal{M}, s \models \psi \\
\mathcal{M}, s & \models & \phi \longrightarrow \psi \text{ iff whenever } \mathcal{M}, s \models \phi \text{ then } \mathcal{M}, s \models \psi \\
\ldots & &
\end{array}
$$

Satisfaction for CTL (c'ed)

Given an FSM $\mathcal{M} = \langle \Sigma, \rightarrow \rangle$, $s \in \Sigma$ and a CTL formula ϕ, then $\mathcal{M}, s \models \phi$ is defined inductively as follows:

$$
\begin{array}{lcl}
\ldots & & \\
\mathcal{M}, s & \models & \mathrm{AX}\,\phi \text{ iff for all } s_1 \text{ with } s \rightarrow s_1 \text{, we have } \mathcal{M}, s_1 \models \phi \\
\mathcal{M}, s & \models & \mathrm{EX}\,\phi \text{ iff for some } s_1 \text{ with } s \rightarrow s_1 \text{, we have } \mathcal{M}, s_1 \models \phi \\
\mathcal{M}, s & \models & \mathrm{AG}\,\phi \text{ iff for all paths } p \text{ with } p_1 = s, \\
& & \quad \text{we have } \mathcal{M}, p_i \models \phi \text{ for all } i \geq 2 \\
\mathcal{M}, s & \models & \mathrm{EG}\,\phi \text{ iff there is a path } p \text{ with } p_1 = s \text{ and} \\
& & \quad \text{we have } \mathcal{M}, p_i \models \phi \text{ for all } i \geq 2 \\
\mathcal{M}, s & \models & \mathrm{AF}\,\phi \text{ iff for all paths } p \text{ with } p_1 = s \\
& & \quad \text{we have } \mathcal{M}, p_i \models \phi \text{ for some } i \\
\mathcal{M}, s & \models & \mathrm{EF}\,\phi \text{ iff there is a path } p \text{ with } p_1 = s \text{ and} \\
& & \quad \text{we have; } \mathcal{M}, p_i \models \phi \text{ for some } i \\
\mathcal{M}, s & \models & \mathrm{A}[\phi\ U\ \psi] \text{ iff for all paths } p \text{ with } p_1 = s \text{, there is } i \\
& & \quad \text{with } \mathcal{M}, p_i \models \psi \text{ and for all } j < i,\ \mathcal{M}, p_j \models \phi \\
\mathcal{M}, s & \models & \mathrm{E}[\phi\ U\ \psi] \text{ iff there is a path } p \text{ with } p_1 = s \text{ and there is } i \\
& & \quad \text{with } \mathcal{M}, p_i \models \psi \text{ and for all } j < i,\ \mathcal{M}, p_j \models \phi
\end{array}
$$

Patterns of Specification

- Something bad (p) cannot happen: AG $\neg p$
- p occurs infinitly often: AG(AF p)
- p occurs eventually: AF p
- In the future, p will hold eventually forever: AF AG p
- Whenever p will hold in the future, q will hold eventually: AG($p \longrightarrow$ AF q)
- In all states, p is always possible: AG(EF p)

LTL and CTL

- We have seen that CTL is more expressive than LTL, but (surprisingly), there are properties which we can formalise in LTL but not in CTL!
- Example: all paths which have a p along them also have a q along them.
- LTL: F $p \longrightarrow$ F q
- CTL: **Not** AF $p \longrightarrow$ AF q (would mean: if all paths have p, then all paths have q), neither AG($p \longrightarrow$ AF q) (which means: if there is a p, it will be followed by a q).
- The logic CTL^* combines both LTL and CTL (but we will not consider it further here).

State Explosion and Complexity

- The basic problem of model checking is **state explosion**.
- Even our small railway crossing has $|\Sigma| = |\Sigma_{Car} \times \Sigma_{Train} \times \Sigma_{Gate}| = |\Sigma_{Car}| \cdot |\Sigma_{Train}| \cdot |\Sigma_{Gate}| = 4 \cdot 4 \cdot 2 = 32$ states. Add one integer variable with 2^{32} states, and this gets intractable.
- Theoretically, there is not much hope. The basic problem of deciding wether a particular formula holds is known as the satisfiability problem, and for the temporal logics we have seen, its complexity is as follows:
  - LTL without U is *NP*-complete.
  - LTL is *PSPACE*-complete.
  - CTL is *EXPTIME*-complete.
- The good news is that at least it is **decidable**. Practically, **state abstraction** is the key technique. E.g. instead of considering all possible integer values, consider only wether i is zero or larger than zero.

Summary

- Model-checking allows us to show to show properties of systems by enumerating the system's states, by modelling systems as **finite state machines**, and expressing properties in temporal logic.
- We considered Linear Temporal Logic (LTL) and Computational Tree Logic (CTL). LTL allows us to express properties of single paths, CTL allows quantifications over all possible paths of an FSM.
- The basic problem: the system state can quickly get **huge**, and the basic complexity of the problem is **horrendous**. Use of abstraction and state compression techniques make model-checking bearable.
- Next lecture: practical experiments with model-checkers (NuSMV and/or Spin)