Linear modeling & verification

Cătălin Dima

LACL, Université Paris 12

Foreword

- The Model-Checking Problem : given a (model of a) program/module/system M and a property P, check whether M satisfies/implements/behaves as required by the property P.
- Ideally, would help a client being convinced that a software provider has produced a solution solving the client's problem.

The model-checking meta-theorem

Model-checking (and in general verification) is about checking/verifying systems for trivial properties.

- That is, properties that can be easily intuited to be true...
- ... but whose verification is tedious, error prone, and event very time consuming on big systems !

Reminder from language theory

- Finite automaton = labeled graph.
- Accepted language = path labels.
  - Starting in initial states, ending in final states.
- Some simple algorithms:
  - Does the automaton accept a given word? (or sequence of labels).
  - Does the automaton have an empty language?
- Some simple constructions:
  - Intersection, complementation, all regexp operations, shuffle.

Modeling systems with automata

- Can be used for modeling any computer system
  - Any realistic system (finite states...).
  - Is rather used for abstracting realistic systems.
- Modeling paradigm : system state = automaton state.
- System step-by-step evolution = transitions.
  - Requires one to identify what is *essential* in a system state to be modeled.
- A shift from transition-labeled to state-labeled automata.

Specifying properties with automata

- Properties too can be “specified” with automata!
- An automaton may represent an intentional aspect
  - A *safety* intention: the system should always keep the value of a variable within some range.
  - A *termination* intention: the program should not run forever, it should reach its final location.
- Properties should utilize system characteristics (variables).
  - In general much simpler than the system model.

Runs

- Run = a sequence of system states.
- What runs accept/generate = sequences of assignments of variables to truth values.
  - The truth value for each variable at time point 0,
  - The truth value for each variable at time point 1,
  - The truth value for each variable at time point 2,
  - ....
- In general, this is a word over the set of atomic propositions AP.
  - $\rho : \mathbb{N} \longrightarrow 2^{AP}$.
- If we want to move to a larger set of atomic propositions, then the states in the automaton need to be expanded.
  - Example...

Model-checking

- So then we have M the system model,
- ... and P the automaton for the property.
- What does it mean for M to satisfy P?
  - All behaviors in M need to satisfy the property P

Model-checking using automata

All words in the language of M need to be also accepted by the automaton P.

- Inclusion between two languages.
- How do we check that, using automata constructions?...

Access control systems

- Subjects, objects, set of rights.
- The matrix of access rights = system state.
- Transitions = commands that change system state.
- Example...
- Runs, accepted words...

Analyzing access control systems

- Safety:
  - Does subject x gain right r onto object y?
- More complicated properties:
  - If a writes to f, then b should never be able to read f.
  - If a reads from f', then a shouldn't be allowed to write to f' after that.
- Trivial to check on a given small-size system, but what if the system is big?
  - *SELinux*: 50.000 lines of code specifying access rights and transitions...
  - Verify it against such a property!
  - *Model-checking (and in general verification) is about checking/verifying systems for trivial properties.*

Specifying access control systems

- An example of an access control system...
- A safety property...
- An integrity property...
- A termination property...
- A property relative to the confinement of the information flow...
- And the result of checking whether property holds within the system...

Scheduling problems

- Suppose we try to implement the mutual exclusion problem with the *strict alternation* protocol:
  - Strict alternation – sharing one variable which shows who's turn is.

while(*true*) {
  while $turn \neq i$ **do** no-op ;
    section critique
  $turn := 1 - i$;
}

- We recall that this is incorrect:
  - What if task 2 loops forever or terminates?

Responsiveness properties

- Once a task is enabled, it should eventually be served.
- Note also that once task 1 is enabled, it remains enabled until it enters in its critical section.
- And that there's nothing said about when the task should stop!
- How do we model that with finite-state automata?...

Infinite words and repeating states

- A Büchi automaton is a finite-state automaton,
- ... but it works on never-ending sequences of labels.
- There is no “final” state, as an infinite word does not have an end!
- There are repeated states F:

Acceptance condition

To accept an infinite word, a run must pass infinitely often through F

- This is equivalent with requiring that the run must pass infintely often through a state from F! (ain’t it?)

Algorithms

- Emptiness?
  - Check whether some repeated state is reachable,
  - ... and reaches itself again!
  - Strongly connected component!
- Intersection?
  - Try to adapt the intersection algorithm from automata over finite words.
  - ... oops! it doesn't work!
  - Can we correct that?

Complementation

- Recall that for complementing, we need deterministic automata.
- Are Büchi automata determinizable?

Proposition

Deterministic Büchi automata are less expressive than nondeterministic ones!

- Try to build a deterministic Büchi automaton for $(a + b)^* b^\omega$.
- Note that $a^* b^\omega$ is accepted by a deterministic Büchi automaton!

Other automata on infinite words

- Need a better notion of determinism.
- Muller automata:
  - A set of sets of repeated states, $\mathcal{F}$.
  - A run is accepting if the set of states states occurring infinitely often is *a member of* $\mathcal{F}$.
- Draw a (deterministic) Muller automaton for
  1. $(a^*b)^\omega$.
  2. $(a+b)^*b^\omega$.
- Do we have $(a+b)^\omega = (a^*b)^\omega$?

Complementation

- Büchi automata can be “transformed” into Muller automata.
- Nondeterministic Muller automata can be “transformed” into Büchi automata.
- Subset construction is not working for Muller automata either.
  - Example
- ... but a modified version (Safra construction) works!
  - Example continued.

Theorem

Büchi The class of languages accepted by Büchi automata is closed under complementation.

- *Exercise:* Rework the intersection construction for Muller automata.

Back to our properties

Büchi/Muller automata for:

- A safety property and its negation.
- An integrity property and its negation.
- A termination property and its negation.
- A property relative to the confinement of the information flow and its negation.
- A responsiveness property and its negation.

Specifying temporal properties

- Büchi automata are nice, graphical representations of properties.
- Algorithmics for them turn into graph algorithmics.
  - Essentially reachability and search for strongly connected components.
  - And various constructions of new graphs from smaller ones.
- It's visual, easy to implement, easy to read, but not very easy to write...
  - It's not easy to guess that an automaton represents a responsiveness property.

Regular expressions

- Equivalent with finite-state automata.
- ω-regular expressions equivalent with Büchi automata.
- Clearly more compact than automata specifications.
- But do we really understand what regular expression mean?
- Write an ω-regular expression for
  - A property of the type p holds forever on.
  - A property of the type p holds until q holds.
  - A property of the type there exists a point where p holds.
- Wouldn't it be possible to have some primitives that correspond to these?

Linear Temporal Logic defined

- Extension of propositional logic.
  - Hence all propositional connectives are present.
- Temporal primitives:
  - Next: $\bigcirc p$.
  - Until: $p \mathcal{U} q$.
  - Globally: Gp or $\Box p$.
  - Forward: Fp or $\Diamond p$.
- Combinations of all these.

Semantics

- Each formula is interpreted over a *run*
  - Or an infinite word, $\rho : \mathbb{N} \to 2^{AP}$.
- Each formula can be interpreted at a time point along the run:

$$
\begin{array}{ll}
(\rho, i) \models p & \text{if } p \in \rho(i) \\
(\rho, i) \models \phi_1 \wedge \phi_2 & \text{if } (\rho, i) \models \phi_1 \text{ and } (\rho, i) \models \phi_2 \\
(\rho, i) \models \neg\phi & \text{if } (\rho, i) \not\models \phi \\
(\rho, i) \models \bigcirc \phi & \text{if } (\rho, i+1) \models \phi \\
(\rho, i) \models \phi_1 \,\mathcal{U}\, \phi_2 & \text{if there exists } j \geq i \text{ with } (\rho, j) \models \phi_2 \\
 & \quad \text{and for all } i \leq k < j, (\rho, k) \models \phi_1
\end{array}
$$

Semantics (2)

- Semantics, continued:

$$(\rho, i) \models \Diamond \phi \text{ if there exists } j \in \mathbb{N} \text{ with } (\rho, j) \models \phi$$
$$(\rho, i) \models \Box \phi \text{ if for any } j \in \mathbb{N}, (\rho, j) \models \phi$$

- But the first modalities are sufficient:

$$\Diamond \phi = \text{true}\, \mathcal{U}\, \phi$$
$$\Box \phi = \neg \Diamond \neg \phi$$

Semantics (3)

- Other operators: new formulas read as follows:
  - $\phi_1 \mathcal{W} \phi_2$: ϕ_1 holds *weakly until* ϕ_2 holds.
  - $\phi_1 \mathcal{R} \phi_2$: ϕ_2 *releases* ϕ_1.
- Semantics:

$$\phi_1 \mathcal{W} \phi_2 = \phi_1 \mathcal{U} \phi_2 \vee \Box \phi_1$$
$$\phi_1 \mathcal{R} \phi_2 = \neg(\neg\phi_1 \mathcal{U} \neg\phi_2) = \phi_2 \mathcal{W}(\phi_1 \wedge \phi_2)$$

From LTL to Büchi automata

- For each formula ϕ, we may build a Büchi automaton A.
- Construction for $\bigcirc p$.
- Construction for $\neg \bigcirc p$.
- Construction for $p \,\mathcal{U}\, q$ and $\neg(p \,\mathcal{U}\, q)$.
  - Better if we work with sets of repeated states.
  - Not exactly like for Muller automata!
  - Each set of repeated states needs to be visited infinitely often.
  - Reducible to Büchi automata (you know how to do it, yes?).
- How to do it in general?

Model-checking algorithm

- Construct the automaton A for $\neg\phi$.
  - Spares a complementation step!
- Intersect A with the automaton for the system.
- Check for emptiness.

Relationship with Büchi automata

- But are LTL and Büchi automata equivalent?
- Büchi automaton for: "p holds at even time points".
  - Caution! p may or may not hold at odd points!
- Can we write an LTL formula for that?...
  - We only can for "p holds at even points and does not hold at odd points"!
- Actually LTL is equivalent with Büchi automata which cannot count!
  - The Büchi automaton for "p holds at even time points" counts modulo 2!

Fixpoints

- Until, weak until, release and the others can be defined “inductively”:

$$\begin{aligned} \Diamond p &\equiv ...? \\ \Box p &\equiv ...? \\ p\,\mathcal{U}\,q &\equiv q \vee (p \wedge \bigcirc(p\,\mathcal{U}\,q)) \\ \neg(p\,\mathcal{U}\,q) &\equiv ...? \end{aligned}$$

- May define least fixpoints and greatest fixpoints
- The “equation” for $p\,\mathcal{U}\,q$ is $X = q \vee (p \wedge \bigcirc X)$.
  - Constructing the solution works by replacing X with false and iterating.
- The “equation” for $\neg(p\,\mathcal{W}\,q)$ is $X = \neg p \wedge (\neg q \vee \bigcirc X)$.
  - Constructing the solution works by replacing X with true and iterating.

Fixpoint LTL

- Utilize only $\bigcirc$ and boolean connectives!
- And two fixpoint operators:
  - μX, least fixpoint, computed starting with $X :=$ false.
  - νX, greatest fixpoint, computed starting with $X :=$ true.
- What does this mean:
  - $\mu X \nu Y \big(p \wedge \bigcirc(X \vee q \wedge Y)\big)$?...
- Not easy to read...
- But equivalent with Büchi automata!

Past time

- Operators refering to the past:
  - Previous: $\bullet p$.
  - Since: $p\,\mathcal{S}\,q$.
  - Always before: $\blacksquare p$.
  - Sometimes: $\blacklozenge p$.
- Write down their semantics on a run!
- Write down their fixpoint equations!

Past normal form

Theorem

Any LTL formula is equivalent with a formula in the following normal form:

$$\Box \Diamond \phi \wedge \Diamond \Box \psi$$

where ϕ and ψ are past formulas.

- Safety properties: $\Box \phi$.
- Termination properties: $\Diamond \phi$.
- Responsiveness properties: $\Box \Diamond \phi$.
- Persistence properties: $\Diamond \Box \phi$.

First-order logic

- Semantics is defined with first-order quantifiers.

$$(\rho, i) \models \phi_1 \mathcal{U} \phi_2 \qquad \text{if there exists } j \geq i \text{ with } (\rho, j) \models \phi_2$$
$$\text{and for all } i \leq k < j, (\rho, k) \models \phi_1$$

- Could we drop temporal operators and use only first-order logic?
  - Logic over integers = positions along a run.
  - Atomic proposition Π_p = sets of positions along a run where p holds.
  - Operators: $\in$, $\leq$, $=$.

First-order logic

- $\Diamond p \equiv \exists i . i \in \Pi_p$
- $\Box p \equiv \forall i . i \in \Pi_p$
- $p \,\mathcal{U}\, q \equiv ...?$
- $p \,\mathcal{S}\, q \equiv ...?$

Theorem (Kamp)

First-order logic of linear time and LTL are expressively equivalent.

Exercise

- Draw an automaton for an easy security protocol.
- Draw an automaton for a confidentiality property for that protocol.
- Verify it!
  - The problem needs to be brought to a finite-state situation.
  - And even then, you further need to simplify it so as to have only very few items (principals, keys, nonces...)!
- *Model-checking (and in general verification) is about checking/verifying systems for trivial properties.*