COL750: Foundations of Automatic Verification (Jan-May 2023)

Lectures 13 & 14 (Transition Systems, Properties, Model Checking[1])

Kumar Madhukar

madhukar@cse.iitd.ac.in

Feb 23rd and 27th

[1] examples used here are from Srivathsan's slides on Model Checking

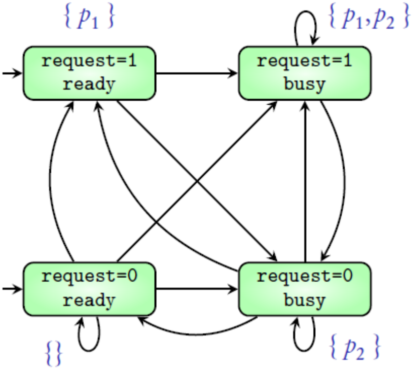
Recall

```
MODULE main

VAR
        request : boolean;
        status : {ready, busy};

ASSIGN

        init(status) := ready;
        next(status) := case
                                request : busy;
                                TRUE : {ready, busy};
                        esac;
```

Transition System

Atomic propositions AP = $\{ p_1, p_2 \}$

p_1: `request=1` p_2: `status=busy`



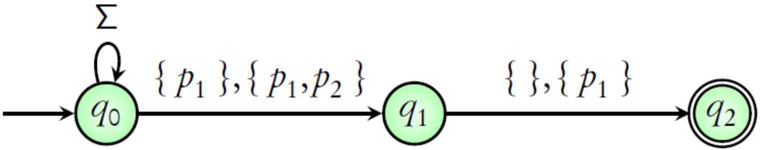
Safety Property

- a property is a set of infinite words over the power-set of atomic propositions
- e.g. p_1 is always true (denotes all those words where each letter is either $\{p_1\}$ or $\{p_1, p_2\}$)
- P is a safety property if there exists a set of bad-prefixes such that P is the set of all words not starting with a bad-prefix
- e.g. if p_1 is true, then p_2 must be true in the next step (the set of bad-prefixes is all those words that have the letter $\{p_1\}$ or $\{p_1, p_2\}$ somewhere, but the immediate next letter is neither $\{p_2\}$ nor $\{p_1, p_2\}$)

Regular Safety Properties

- a safety property is called a regular safety property if the set of bad-prefixes is a regular language (can be recognized by an NFA)

$$\Sigma = \{ \{\}, \{p_1\}, \{p_2\}, \{p_1, p_2\} \}$$



Regular Safety Properties

- not all safety properties are regular safety properties
- e.g. consider the property that at any point, the total number of occurrences of p_1 so far must exceed the total number of occurences of p_2
- a bad-prefix is a word that has fewer p_1's than p_2's
- the set of bad-prefixes is not a regular language

Invariants

- properties of the form "ϕ is always true" (or, $\mathtt{G}\ \phi$)
- where ϕ is a boolean expression over the atomic propositions
- it is easy to see that invariants are regular safety properties

Checking properties in transition systems

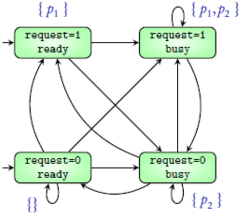
- How can we check if a given transition system satisfies an invariant property?
  - every reachable state must satisfy the property
  - depth-first search
  - it is useful to obtain a counterexample if the property is violated
  - the dfs can also be modified to print the entire path to the violating state (instead of just reporting the violating state)
- What about regular safety properties?
  - take the synchronous product of the transition automaton and the bad-prefix automaton, and check if it's language is non-empty.

Checking regular safety properties

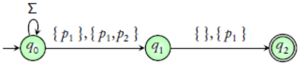
Model

Atomic propositions AP = $\{ p_1, p_2 \}$

p_1: `request=1` p_2: `status=busy`



Safety property



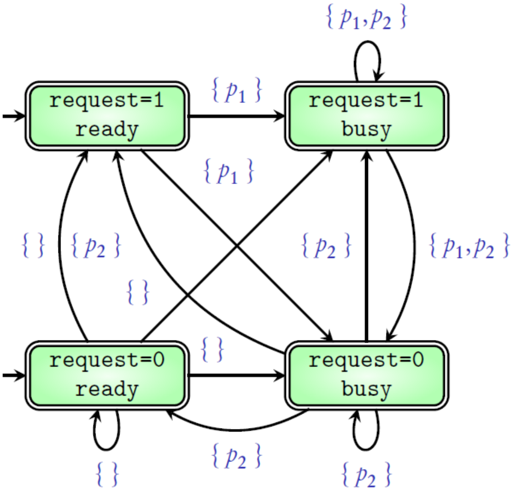
BadPrefixes

Does the model satisfy the safety property?

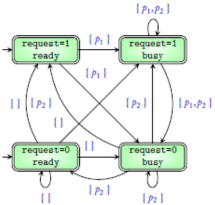
Transition System $\rightarrow$ Automaton

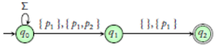
- move the labels from the states to all the outgoing transitions from that state
- make every state an accepting state

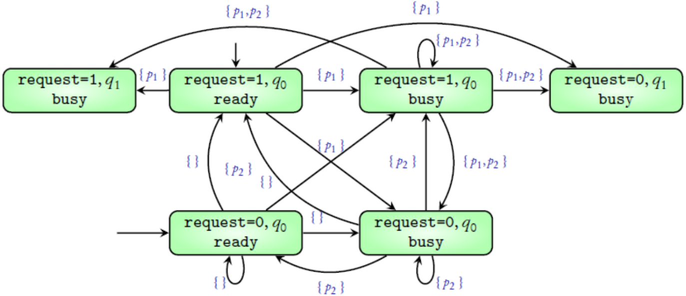
Transition System → Automaton



Emptiness check on Product automaton







Exercises from the last class

While proving that a language is Büchi-recognizable iff it is ω-regular, we had left the following two claims as an exercise.

1. If U is regular, then U^ω is Büchi-recognizable.
2. If U is regular, and L is Büchi-recognizable then UL is Büchi-recognizable.

We can show this by explicitly constructing an NBA using the NFA for U and the NBA for L.

Here's the reference material (see slides 6–19) for this construction:
https://www.cmi.ac.in/~sri/Courses/NPTEL/ModelChecking/Slides/Unit6-Module2.pdf

LTL Model Checking

Here's the reference material for this part:

https://kumarmadhukar.github.io/courses/verification-holi23/resources/ltl-mc-srivathsan.pdf

Thank you!