IBM

The Safety Simple Subset

Sitvanit Ruah

Joint work with
Shoham Ben-David and Dana Fisman

IBM

Outline

- An introduction to PSL and the simple subset
- Overview of the contribution
- Technical part

IBM

PSL

- A property specification language
- An IEEE standard
- Widely used in industry both for simulation and for model checking

IBM

Motivation

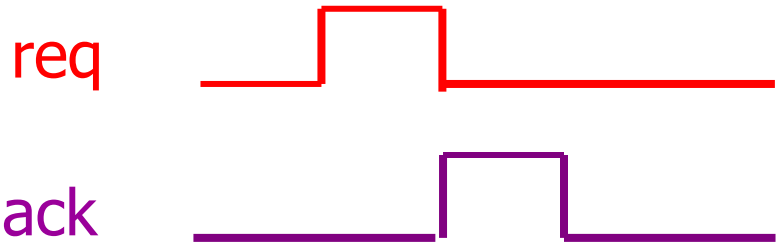
- There is a growing interest in PSL and in **efficient** algorithms for PSL
- There are independent results in the literature for related languages but no **handy** algorithm that combines them all for PSL implementers
- The PSL LRM defines the simple subset which intuitively should have efficient algorithms for simulation

IBM

The Simple Subset

- In the **simple subset**, time advances monotonically, left to right through the property, like in a timing diagram

always (*req* **->** **next** *ack*) is in the simple subset



always ((**next** *ack*) **->** *req*) is not in the simple subset

IBM

Motivation CONT.

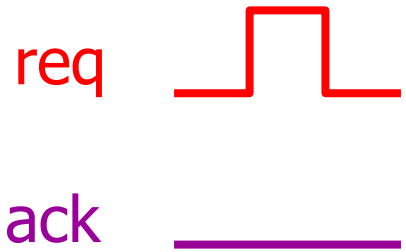
- The PSL LRM does not provide a construction or any other proof that verification of properties in the simple subset is more efficient

IBM

The Safety Simple Subset

- A formula f is a safety formula if it has a finite counter example

- always (*req* **->** next *ack*) is a safety formula
- A counter example for this formula:



always (*req* **->** eventually! *ack*) is not a safety formula

IBM

The Safety Simple Subset

- Safety properties can be reduced to invariance checking
- Therefore they are easier for model checking
- We focus on the safety simple subset because we are interested both in simulation and in model checking

IBM

RLTL^{LV}

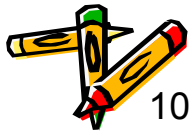
- PSL was defined for practical applications
- It is a rich language with many syntactic sugaring operators
- In this work we focus on the core operators for the theoretical analysis
- We define RLTL^{LV} –

  the subset of linear violation
- There exists an automaton on finite words of linear size detecting violation of properties in RLTL^{LV}
- The exact relation to the safety simple subset defined in the LRM will be described later

IBM

Contribution

- Defining a subset with the most efficient (linear) automaton on finite words
  - giving guidelines for verification engineers on writing efficient properties
- Providing automata constructions for properties in $RLTL^{LV}$
  - Can be directly used by tool implementers
- Relating independent previous results on languages that are subsets of PSL



IBM

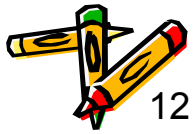
Technical Outline

- Definition of $RLTL^{LV}$ –
  the subset of linear violation
- Construction via a violating RE
- Examples of the direct automata construction
- Comparison to related results
- Conclusions

IBM

Regular Expressions (REs)

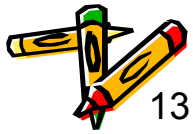
- A boolean expression b is a regular expression
- If r, r1 and r2 are regular expressions then so are the following
  - r1;r2 concatenation
  - r1 | r2 or
  - r[*] consecutive repetition
  - r1 : r2 fusion ({a;b;c}:{d;e} == {a;b;c&d;e})



IBM

The Subset of Linear Violation ($RLTL^{LV}$)

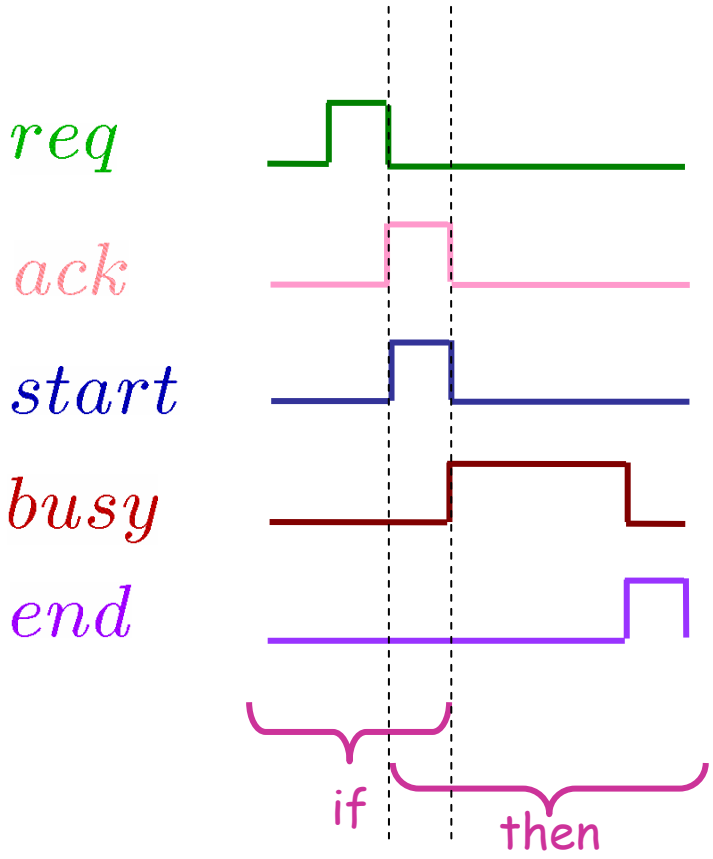
- Let b be a boolean, r a regular expression and f, f1,f2 $RLTL^{LV}$ formulas. The following are in $RLTL^{LV}$:
  - b
  - f1 & f2
  - next f (same as X f)
  - (b & f1) | (!b & f2)
  - (b & f1) until (!b & f2)
  - {r} |-> f



IBM

{r}|->f

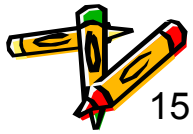
{[*]; req; ack} |-> (start & next (busy until end))



IBM

Automata Construction

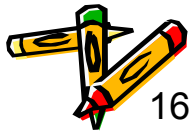
- The restriction to safety allows the use of automata on finite words
- We construct an NFA recognizing the violation of the given formula
- The bad states - the accepting states of the NFA
- The verification problem is reduced to model checking the invariance "the NFA is not in a bad state" on a parallel composition of the model with the NFA



IBM

Automata Construction – cont.

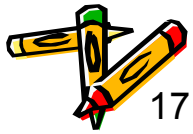
- We give two constructions:
  - Via a violating RE – first build an RE and then construct an automaton recognizing its language
  - A direct automata construction



IBM

Construction via a Violating RE – V(f)

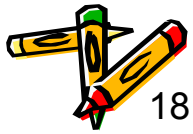
- For a formula f in $RLTL^{LV}$ we construct an RE V(f) recognizing violation of f
  - V(b) == !b
  - $V(f_1 \,\&\, f_2) == V(f_1) \mid V(f_2)$
  - V(next f) == true; V(f)
  - $V(b \,\&\, f_1 \;\mid\; !b \,\&\, f_2) == \{b{:}V(f_1)\} \;\mid\; \{!b : V(f_2)\}$
  - $V((b \,\&\, f_1) \text{ until } (!b \,\&\, f_2)) == b[*];\{b{:}V(f_1)\} \;\mid\; \{!b{:}V(f_2)\}$
  - V({r} |-> f) == {r}:V(f) (f is violated at the last cycle of a prefix satisfying r)



IBM

Construction via a Violating RE – V(f)

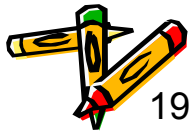
- [[r!]] - the set of finite/infinite words that have a finite prefix matching r
- Theorem: |V(f)| = O(|f|) and [[V(f)!]] recognizes violation of f
- Given the RE V(f) there exists a finite automaton of linear size accepting [[V(f)!]]
- Can use existing NFA constructions for REs



IBM

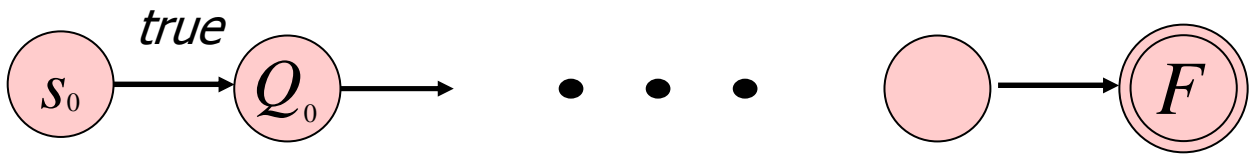
Direct Automata Construction

- The direct construction resembles the violating RE construction
- A direct construction enables optimizations at the automata level



A Simple Example – X f

- V(X f) = {true;V(f)}

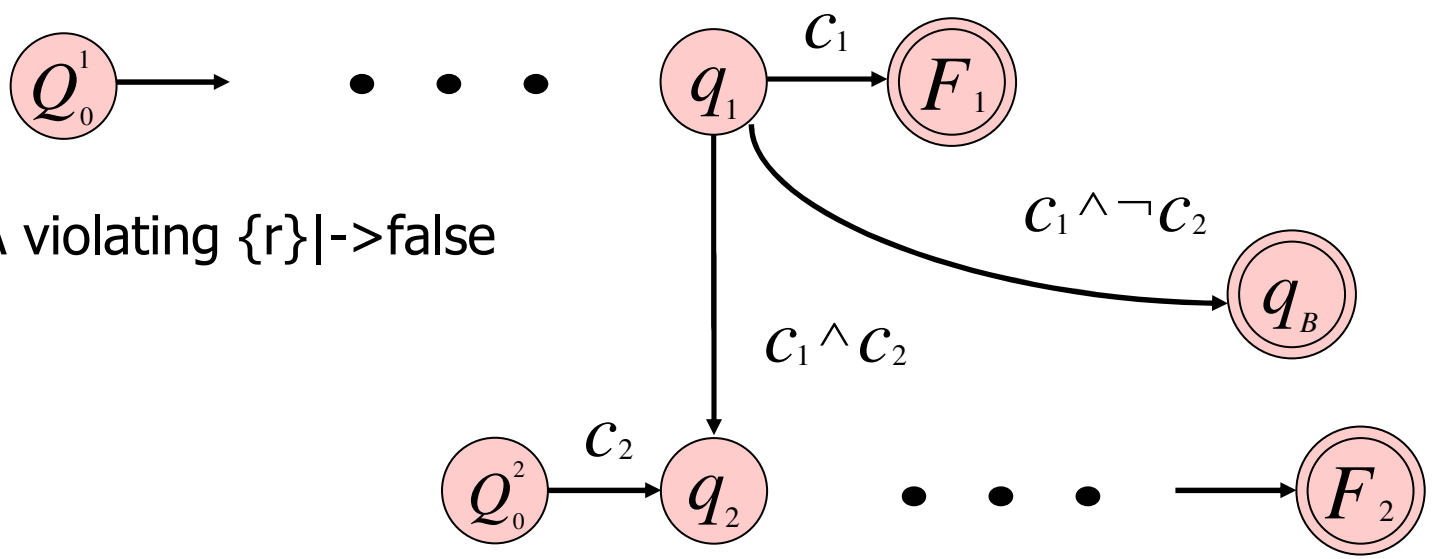


NFA recognizing violation of f

NFA recognizing violation of X f

{r} |-> f

- V({r}|->f) = {r}:V(f)



NFA violating {r}|->false

NFA violating f

NFA violating {r}|->f

IBM

RLTL^{LV} vs. the Safety Simple Subset

- The safety simple subset contains weak RE:
  - {a;b[*];c} holds on a path iff we never fall off the automaton for {a;b[*];c}
  - For a general weak RE the size of the automaton is **exponential** (deterministic)
- For simplicity reasons, the restrictions in the LRM on **until** and | are stronger than in RLTL^{LV}

IBM

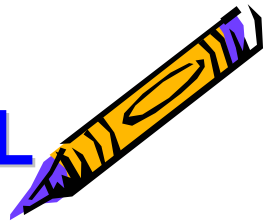
Classification of Safety Formulas

- [KV99] studied the complexity of model checking safety formulas
- The most efficient automata construction they introduced was for non-pathologically safe formulas
- They showed that violation of these formulas can be detected by a non-deterministic automaton on finite words of exponential size
- Every f in $RLTL^{LV}$ is non-pathologically safe
- Violation of $RLTL^{LV}$ formulas can be detected by a non-deterministic automaton on finite words of linear size

$RLTL^{LV}$ vs. the Common Fragment of CTL and LTL

- Maidl [00] studied the subset of ACTL formulas that have an equivalent in LTL
- She gave a syntactic definition of the common fragment of ACTL and LTL - LTL^{det}
- We observed that the LTL part of the PSL simple subset meets the requirements of LTL^{det}
- We extend safety LTL^{det} with regular expressions
- Safety LTL^{det} is strictly less expressive than $RLTL^{LV}$ (Wolp83)

RLTL^{LV} vs. On-the-fly Model Checking RCTL



- BBL[98] defined the logic RCTL that extends CTL with regular expressions
- They studied on-the-fly model checking - invariance checking on a parallel composition of the model with a finite automata
- Defined RCTL^{OTF} - can be verified on-the-fly
- We extended it to RCTL^{LV}
- RLTL^{LV} is the subset of PSL expressively equivalent to RCTL^{LV}

IBM

Conclusions

- Previous results:
  - Studied subsets of LTL and CTL mainly from an expressibility point of view [Mai00],[KV99]
  - Addressed larger subsets (of LTL) thus the resulting algorithms were less efficient [Mai00],[KV99]
  - Did not treat regular expressions [Mai00],[KV99]
  - Addressed a smaller subset [BBL98]
- Our work:
  - Extends previous results in order to apply them to PSL
  - Refines previously defined subsets to get more efficiency, that is, a linear finite automata