Bounded Model Checking of Temporal Formulas with Alloy

Alcino Cunha

HASLab
HIGH-ASSURANCE SOFTWARE LABORATORY
INESCTEC

Universidade do Minho

ABZ 2014, Toulouse

Motivation

- Alloy is great for inductive (indirect) analysis of safety and liveness properties.

Motivation

- Alloy is great for inductive (indirect) analysis of safety and liveness properties.

C. Newcombe: *Why Amazon Chose TLA+*.

We have tried the third technique (inductive invariance), but **have found it difficult to apply to real-world systems** because we don't yet have good tools to help discover complex inductive invariants.

Motivation

- Alloy is great for inductive (indirect) analysis of safety and liveness properties.

C. Newcombe: *Why Amazon Chose TLA+.*

We have tried the third technique (inductive invariance), but **have found it difficult to apply to real-world systems** because we don't yet have good tools to help discover complex inductive invariants.

- Can we perform direct analysis with Alloy?
  - There exists a popular *trace idiom* that enables *bounded model checking* of safety properties.
  - But what about liveness properties?

Motivation

D. Jackson: *Software Abstractions*.

Instead [of inductive analysis], we'll take an approach that requires less insight, **and allows the** [safety and liveness] **properties** [of a leader election protocol] **to be checked directly.**

Motivation

D. Jackson: *Software Abstractions*.

Instead [of inductive analysis], we'll take an approach that requires less insight, **and allows the** [safety and liveness] **properties** [of a leader election protocol] **to be checked directly.**

- In fact, applying the trace idiom to check liveness properties yields spurious counter-examples.
- In this leader election protocol case study, some indirect analysis was still used to rule out such counter-examples.

Motivation

P. Zave: *A Practical Comparison of Alloy and Spin.*

There are no temporal operators in Alloy. Strictly speaking the progress property could be expressed in Alloy using quantification over timestamps, but there is no point in doing so because the Alloy Analyzer could not check it meaningfully... **For all practical purposes, progress properties cannot be asserted in Alloy.**

Motivation

- Although not pursued in *Software Abstractions*, the book already points to a solution that would allow proper direct analysis.

Biere et al: *Symbolic model checking without BDDs*.

We introduce a bounded model checking procedure for LTL which reduces model checking to propositional satisfiability.

Motivation

- This paper shows how bounded model checking can be done with Alloy:
  - Uses a simple variant of the trace idiom.
  - Trivial to adapt existing Alloy models using the trace idiom to use it instead.
  - LTL formulas can be automatically translated to Alloy, but simple liveness properties can also be asserted directly.
  - Preliminary evaluation shows that direct analysis is doable, provided trace prefixes are short.

The trace idiom by example

```
open util/ordering[Process]

sig Time {}

sig Process {
  succ: Process,
  toSend: Process -> Time,
  elected: set Time
}

fact ring {
  all p: Process | Process in p.^succ
}
```

The trace idiom by example

```
pred init [t: Time] {
  all p: Process | p.toSend.t = p
}

pred step [t, t': Time, p: Process] {
  let from = p.toSend, to = p.succ.toSend |
    some id: from.t {
      from.t' = from.t - id
      to.t' = to.t + (id - p.succ.prevs)
    }
}

pred skip [t, t': Time, p: Process] {
  p.toSend.t = p.toSend.t'
}
```

The trace idiom by example

```
open util/ordering[Time]

fact traces {
  init [first]
  all t: Time - last | let t’ = t.next |
    all p: Process |
      step [t, t’, p] or
      step [t, t’, succ.p] or
      skip [t, t’, p]
}
```

The trace idiom by example

```
assert AtMostOneElected {
  lone elected.Time
  -- all t : Time | lone elected.t
}

pred progress  { all t: Time, t' : t.next |
    some Process.toSend.t =>
      some p: Process | not skip [t, t', p]
}

assert AtLeastOneElected {
  progress => some elected.Time
  -- progress => some t : Time | some elected.t
}
```

The trace idiom by example

- `check AtLeastOneElected for` n `Process,` t `Time` yields counter-examples for $n = 3 \wedge t < 7$, $n = 4 \wedge t < 11$, etc.
- For such simple property and protocol, it is trivial to rule out such counter-examples as spurious (not enough time to finish protocol. . .).
- For more complex properties and systems such (still indirect) reasoning may be far from trivial.

Bounded model checking with Alloy

- Bounded model checking prescribes that a prefix of a trace is a valid counter-example of a liveness property if it contains a *back loop* in the last state.
- Valid counter-examples are thus finite representations of infinite traces.

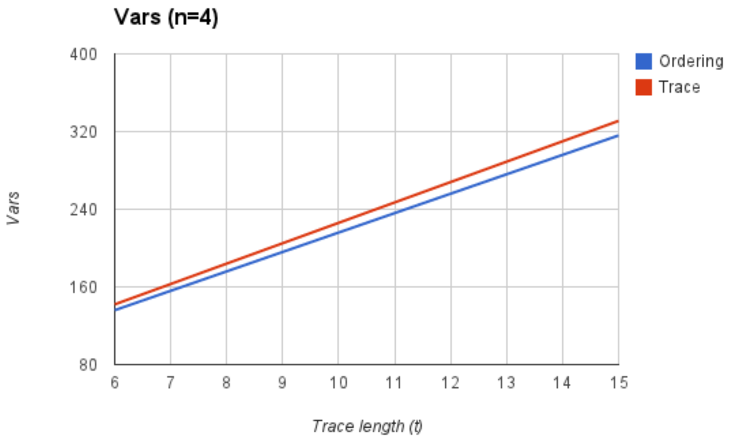
Bounded model checking with Alloy

- Bounded model checking prescribes that a prefix of a trace is a valid counter-example of a liveness property if it contains a *back loop* in the last state.
- Valid counter-examples are thus finite representations of infinite traces.
- It is easy to define a parameterized `trace` module with the same interface as `util/ordering`, but that allows a back loop in the last state:
    - It defines a predicate `infinite` that checks if the loop is present in a trace instance.
    - Dually for `finite`.
    - When `infinite` holds relation `next` assigns a successor to every state, thus modeling an infinite trace.

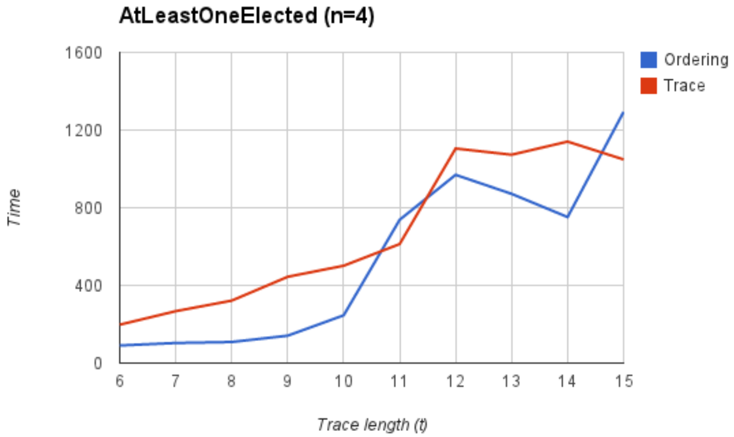
Back to our example

```
--open util/ordering[Time]
open trace[Time]

fact traces {
  init [first]
  -- all t: Time - last | let t' = t.next |
  all t: Time, t' : t.next |
    all p: Process |
      step [t, t', p] or
      step [t, t', succ.p] or
      skip [t, t', p]
}

assert AtLeastOneElected {
  progress => finite or some elected.Time
}
```

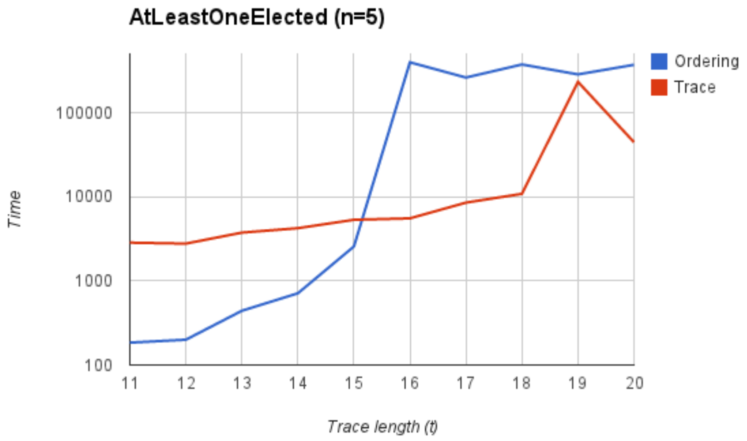
Evaluation


Vars (n=4)
400
320
240
160
80
Vars
6
7
8
9
10
11
12
13
14
15
Trace length (t)
Ordering
Trace

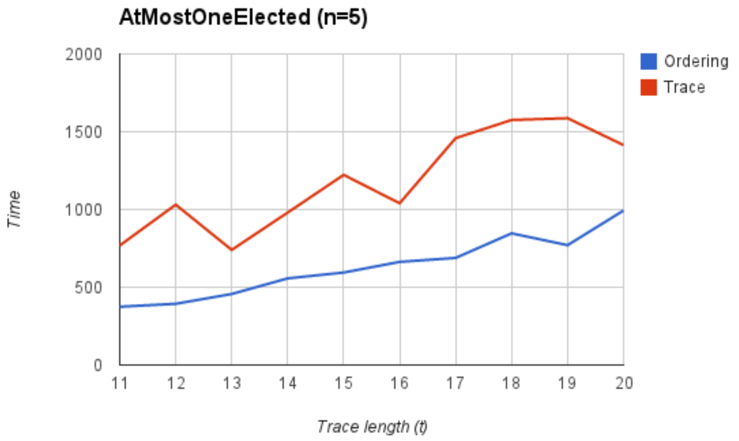
Evaluation


AtLeastOneElected (n=4)
Time
Trace length (t)
1600
1200
800
400
0
6
7
8
9
10
11
12
13
14
15
🟦 Ordering
🟥 Trace

Evaluation: log scale!



Evaluation


AtMostOneElected (n=5)
2000
1500
1000
500
0
Time
11
12
13
14
15
16
17
18
19
20
Trace length (t)
Ordering
Trace

Conclusion

- Liveness properties can be asserted and verified directly (with bounded model checking) in Alloy.
- Penalty of doing proper bounded model checking can be negligible. Can even be more efficient!
- In general, it seems feasible for short trace prefixes.
- Trivial to adapt existing Alloy models based on the trace idiom to do proper bounded model checking.