Using how to “read a paper” by S. Kshav as a guide

First pass

* Carefully read
  + Title: the Model checker spin(not much here)
  + Abstract
    - Efficient verification for distributed software systems
      * Detect design errors
      * Works on high level modeling
      * Works on detailed code modeling
    - Overview of design, structor, theoretical foundation, of verifier
    - Overview of example applications
  + Introduction
    - Generic support for verification of asyncrouns systems
    - Specificationwith
      * Randezwous primitives
      * Asynchronous message passing through buffered channels
      * Shared variables
      * Combinitions of above
    - Distingueses itself with focus on asynch software instead of sync in hardware
    - Main goals
      * Aimes
        + Intuitive
        + program like notation
        + Unambiguous
        + Details not neccicary (does this mean higher level not needing lower?)
      * Powerful concise notation for correctness requirements
      * Methodology for achieving aims and matching them with correctness verification
    - Structure
      * Start with high level specification of the model or algorithm
        + Spin checks for syntax errores which you fix
      * Use interactive till you feel things work as you want
      * Spin is used to generate omtimized on the fly verification program from high level specifications(what does this mean)
      * Runes the verification and spits out a counter example if applicable
* Read section and subsection headings but nothing else
  + Foundation
    - Temporal logic requirements
    - Domain of Application
  + Algorithms
    - Nested Depth-first Search
    - From LTL Forula to Buchi Automata
    - Partial Order Reduction
    - Memory Management
      * State Compression
      * Bit-State Hashing
  + Practical Applications
    - Process Scheduling
    - Leader Election
      * A modification
    - Flow Control
      * Safety Properties
      * Liveness Properties
    - Other Application
* Read conclusion
  + Scarce capapbilites for conccuurrent systems
  + Still build on the tools for this from before
  + Design methodology as follows
    - Distinction from behavor and requirements on behavior
      * Unambiguously defining the two aspects in a verification prototype in PROMELA
    - Prototype is verified using spin. Both requirements and behavier are checked for internal and mutal consistency.

Answer

* Category(measurement, analysis, description, prototype)
* Context (related papers)
* Correctness
* Contributions
* Clarity
  + Certain sections especially on pg 4 weren’t the clearest

Needed for other presentation.

Determan

* Background info
  + It pulls from a deep and rich ppol of resources from previous model checking stuffs
* Motivation(problem being soved or question answered)
  + Modeling system for distributed software systems
  + Generic support for verification of asyncrouns systems
  + Notation that follow aims as above
    - Promila used
  + Expressive notation for correctness
    - LTL
      * Technically wouldn’t it be LTLf since it can only work on finite traces?
* Products of the paper(specifically novel), tools, solutions, artifacts
  + A tool for “Formally” verifying Dystributed systems
* Reproducable
  + When I went to redo this for myself, I didn’t get quite the same results
    - See pg 4 fig 3.
    - Scheduler’s DFA is basically the same but has 2 extra states though these states seem to automatically be moved past.
    - XSpin isn’t the main used graphical front end however, all the parts here where shown through the terminal.
* Is it correct
  + It is at least correct that if it shows an error there is an error.
  + Puhaps this is more of a user issue
  + IT seems to either have issues with communication, I am misreading, or I’m reading it wrong.
  + Depending on the distributive system, it may be pulling from an infinite state automata(page 10 of Distributed Computing book) though it mentiones here that processes can only have finitly many control states
* Buidable(exdent or utilize ofr different projects)
  + Absolutely we have already seen other work with generic concurrent system rather than just distributed systems
* Future work
  + Some has already been done with updating the UI, There are also are probably some other refinements of the various efficiency algorithms
  + Build tools to take a distributed model and auto generate the promela

Cited in at least 2,548 papers

A computer screen shot of a diagram

Description automatically generatedA blue screen with black circles and yellow text

Description automatically generated

A screenshot of a computer code

Description automatically generated

Motivation

* + This was not the start of spin. Spin first 1991 and this is in 1997
  + Modeling system for distributed software systems
  + Distributed tools in it’s infency
  + Generic support for verification of asyncrouns systems
  + Have good support for verification
  + Notation that follow aims as above
    - Promila used
  + Use various methods to more efficiently find out how efficiently a system can be run

Foundation:

* Process templates are translated by spin into finite automaton
  + Each process can instantiate other processes
  + Global is constructed by computing an asynchronous interleaving product of these automata (creating a new automata)
* Go to next slide
* As aware LTL as Buchi automata but a negation of this Buchi automata is more useful
* Now takes the product of the Buchi automata and the Globally constructed automata (creates a Buchi automata).
* A non empty resulting Buchi automata shows that there are runs that satisfy the LTL claims however remvmber that our Buchi automata represents a NEGATIVE claim
* This is tested with a nested Depth First search depth first search looking for an acceptance cycle
  + Notes on nested Depth First search
    - Reduction tyechnick
    - Doesn’t show all cycles like Tarjan’s
    - Will detect AT LEAST ONE if a cycle exists which is good enough
      * Can be extended with weak fairness with Choueka’s flag construction (look at references)
* Know it’s empty if the complete intersesection product has been computed

Uses

* General Distributed systems
* Verifying Concurrent algorithms
* Specific examples of
  + Scheduling
  + Leader election
  + Flow Controll
* KEEP SAME SLIDE Future: thanks to freely given
  + general usage in studying theoretical frameworks
  + empirical studes on relative various algorithms
  + Later Extensions and revitions
  + As already seen, used for proactical application for model checking

Issues:

Don’t forget to mention other weirdness with confusing wordings

167/196=