An Incremental Approach to Model Checking Progress Properties

Aaron Bradley    Fabio Somenzi    Zyad Hassan    Yan Zhang

Department of Electrical, Computer, and Energy Engineering
University of Colorado at Boulder

FMCAD, 1 November 2011
Outline

1. Introduction
2. The FAIR Algorithm
3. Experiments
4. Conclusions
Outline

1. Introduction
2. The FAIR Algorithm
3. Experiments
4. Conclusions
Property Classification

Reactivity

Recurrence

Persistence

Obligation

Safety

Guarantee

Linear Time Hierarchy

Safety: IC3
Progress: FAIR over IC3
Generalized Büchi Automata

- Given:
  - Fair Transition System (FTS) $S$
  - LTL property $P$

- Compute **generalized Büchi automaton** $C = \mathcal{A}_{\neg P} \parallel S$.

- If $S$ is finite state, nonemptiness of $C$ corresponds to the existence of a **reachable fair cycle**, aka **lasso**.
A lasso’s cycle is contained in a **strongly connected component** (SCC) of the state graph.

A nonempty set of states is **SCC-closed** if every SCC is either contained in it or disjoint from it.

A partition of the states into SCC-closed sets is a coarser partition than the SCC partition; hence, ... 

Every cycle of a graph is contained in some SCC-closed set.
Outline

1. Introduction
2. The FAIR Algorithm
3. Experiments
4. Conclusions
Reachable Fair Cycles

Reduce search for reachable fair cycle to a set of safety problems:

- **Skeleton:**

  States of skeleton together satisfy all fairness constraints.

- **Task:** Connect states to form lasso.
Reach Queries

Each connection task is a reach query.

- **Stem query**: Connect initial condition to a state:

- **Cycle query**: Connect one state to another:

(To itself if skeleton has only one state.)
Witness to Nonemptiness

If all queries are answered positively:

Witness to nonemptiness of $C$. 
Global Reachability

If a stem query is answered negatively: new **inductive** global reachability information.

- Constrains subsequent selection of skeletons.
- Constrains subsequent reach (stem and cycle) queries.
- Improve proof by strengthening (using ideas from IC3).
Barriers: Discovering SCC-Closed Sets

If a cycle query is answered negatively: new information about SCC structure of state graph.

- Inductive proof: “one-way barrier”
- Each “side” of the proof is SCC-closed.
- Constrains subsequent selections of skeletons: all states on one side.
Using Barriers for Generalization

- Can be used to constrain subsequent cycle queries.
  - Not necessary for completeness.
  - Can increase IC3’s generalization power.
  - But can negatively impact SAT solver.
  - Must choose carefully which barriers to use.
- Improve proof by making smaller (using ideas from IC3).
Key Insights

- **Inductive assertions** describe **SCC-closed sets**.
- **Arena**: Set of states all on the same side of each barrier.
- Unlike previous symbolic methods:
  
  **Barrier constraints on the transition relation combined with the over-approximating nature of IC3 enable the simultaneous (symbolic) consideration of all arenas.**

- A proof can provide information about many arenas even though the motivating skeleton comes from one arena.
Methodological Parallels with IC3

<table>
<thead>
<tr>
<th>IC3</th>
<th>FAIR</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Seed:</strong> CTI</td>
<td>Skeleton</td>
</tr>
<tr>
<td><strong>Lemma:</strong> Inductive clause</td>
<td>Global reachability proof One-way barrier</td>
</tr>
<tr>
<td></td>
<td><em>Relative to previously discovered lemmas.</em></td>
</tr>
<tr>
<td><strong>CEX:</strong> CTI sequence</td>
<td>Connected skeleton</td>
</tr>
<tr>
<td></td>
<td><em>Discovery guided by lemmas. Not minimal.</em></td>
</tr>
<tr>
<td><strong>Proof:</strong> Inductive strengthening</td>
<td>All arenas skeleton-free</td>
</tr>
<tr>
<td></td>
<td><em>Sufficient set of lemmas.</em></td>
</tr>
</tbody>
</table>
Motivating example: $n$-bit counter

- Latches: $b_0, \ldots, b_{n-1}$ (least- to most-significant)
- Output: $o$ switches to 1 and stays when all $b_i = 1$
- Initially: all 0
- Fairness condition: infinitely often $o = 0$

Unfair: after first rollover, henceforth $o = 1$. 

![Diagram of state transitions](attachment:state_transitions.png)
Ideal Proof

First barrier: \( o \)
- Inductive because once \( o = 1 \), it stays 1
- No skeletons among \( o \)-states
- Constrain cycle queries: \( \neg o \land \neg o' \)
Ideal Proof

Second barrier: $b_{n-1}$

- Inductive relative to $\neg o$
- Once $b_{n-1} = 1$, it stays 1 in the $\neg o$-arena
- Both sides have skeletons
- Constrain cycle queries: $b_{n-1} \leftrightarrow b'_{n-1}$
Ideal Proof

Third barrier: $b_{n-2}$

- Inductive relative to previous barriers
- Once $b_{n-2} = 1$, it stays 1 in every arena defined by the previous barriers
- Both sides have skeletons in at least one arena
- Constrain cycle queries: $b_{n-2} \leftrightarrow b'_{n-2}$

And so on. Proof is linear in size of model.
Skeleton-Independent Proofs

- Only a lucky sequence of skeletons would yield ideal proof.
- Therefore: periodically test given predicates, such as single literals, to see if they are barriers (relative to current information).
- A predicate that is not an inductive barrier at one point can become inductive with new information.
Characteristics of FAIR

- Property directed (except skeleton-independent proofs)
- Relies on IC3, thus capitalizes on its strengths
- With IC3, approximating/abstracting
- Highly parallelizable even beyond IC3
Outline

1. Introduction
2. The FAIR Algorithm
3. Experiments
4. Conclusions
Experiments

- Evaluation on 30 models from 9 families
  - Contributed to the HWMCC11 benchmark set
  - Some from literature, most of which contrived
  - Most from VIS benchmark set
  - Number of fairness constraints ranges from 1 to 33
- Four different settings of FAIR considered
- Results compared to those of six other methods
  - Three BDD-based methods: GSH, Lockstep, D’n’C
  - Three variations of the liveness-to-safety scheme
FAIR Compared to GSH
FAIR Compared to D’n’C
FAIR Compared to LTS/IC3

![Graph showing the comparison between FAIR and LTS/IC3 times. The x-axis represents LTS/IC3 times, and the y-axis represents FAIR times. The graph is in a log-log scale, with markers indicating the performance of each method. The trend line shows a linear relationship.]
Results in Summary

- FAIR solved 27–28 problems out of 30 (depending on variation)
- GSH, D’n’C, LTS/IC3 solved 21 problems each
- LTS/ABC solved 20 problems
- Lockstep suffers when there are many SCCs (solved 12 problems)
- LTS/ITP solved 9 problems
Outline

1. Introduction
2. The FAIR Algorithm
3. Experiments
4. Conclusions
Going Forward

- Selection of skeletons
- Proof improvement
- Deciding when to use a barrier to constrain cycle queries
- SAT solver: efficient handling of DNF
- SAT solver: highly incremental
- Distributed implementation
- Integrating BDDs
Conclusions

FAIR: a new approach to SAT-based LTL model checking
- In fact, to model checking all $\omega$-regular properties
- Discovery of SCC-closed sets via safety queries
- One-way barriers: (relatively) inductive assertions
- Property-focused, approximating
- Not only uses IC3 but also follows its principles