

Bounded Model Checking with Fixpoints

Alan Frisch, Daniel Sheridan and Toby Walsh
{frisch, djs, tw}@cs.york.ac.uk
<http://www.cs.york.ac.uk/~djs>

Artificial Intelligence Group
Department of Computer Science, University of York
York YO10 5DD, United Kingdom

Fixpoint Characterisations

Each temporal logic operator is equivalent to the fixed point of a function:

$$Z = \tau(Z)$$

We write $\mu Z.\tau(Z)$ for the *least fixpoint* and $\nu Z.\tau(Z)$ for the *greatest fixpoint* of τ .

$$\begin{aligned}\mathbf{F} f_1 &= \mu Z.f_1 \vee \mathbf{X} Z \\ \mathbf{G} f_1 &= \nu Z.f_1 \wedge \mathbf{X}_1 Z \\ [f_1 \mathbf{U} f_2] &= \mu Z.f_2 \vee (f_1 \wedge \mathbf{X} Z) \\ [f_1 \mathbf{R} f_2] &= \nu Z.f_2 \wedge (f_1 \vee \mathbf{X} Z)\end{aligned}$$

Tarski's Theorem

A fixpoint can be computed by repeatedly applying τ to the maximum value of Z (for the least fixpoint) or the minimum value of Z (for the greatest fixpoint)

The Separated Normal Form

A series of transformations based on fixpoints converts an LTL expression into a conjunction of rules:

$$\mathbf{G} \left(\bigwedge_i (P_i \Rightarrow F_i) \right)$$

Each $P_i \Rightarrow F_i$ can be

$$\begin{aligned} \mathbf{start} &\Rightarrow \bigvee_j l_j && \text{An initial rule} \\ \bigwedge_i l_i &\Rightarrow \mathbf{X}_1 \bigvee_j l_j && \text{A global } \mathbf{X}_1\text{-rule} \\ \bigwedge_i l_i &\Rightarrow \mathbf{X} \bigvee_j l_j && \text{A global } \mathbf{X}\text{-rule} \\ \bigwedge_i l_i &\Rightarrow \mathbf{F} \bigvee_j l_j && \text{A global } \mathbf{F}\text{-rule} \end{aligned}$$

and where l_i and l_j are literals.

Example transformations:

$$\begin{aligned} T_G(\{P \Rightarrow \mathbf{G} f\} \cup \Psi) &= \left\{ \begin{array}{l} P \Rightarrow f \wedge x \\ x \Rightarrow \mathbf{X}_1 (f \wedge x) \end{array} \right\} \cup \Psi \\ T_U(\{P \Rightarrow f \mathbf{U} g\} \cup \Psi) &= \left\{ \begin{array}{l} P \Rightarrow g \vee (f \wedge x) \\ x \Rightarrow \mathbf{X} (g \vee (f \wedge x)) \\ P \Rightarrow \mathbf{F} g \end{array} \right\} \cup \Psi \\ T_{ren1}(\{P \Rightarrow \mathbf{G} f(\mathbf{F} g)\} \cup \Psi) &= \left\{ \begin{array}{l} P \Rightarrow \mathbf{G} f(x) \\ x \Rightarrow \mathbf{F} g \end{array} \right\} \cup \Psi \end{aligned}$$

The Fixpoint Normal Form

SNF only transforms least fixpoint operators: *eventually* (**F**) is left unchanged. We add an extra transformation:

$$T_F(\{P \Rightarrow \mathbf{F} f\} \cup \Psi) = \left\{ \begin{array}{l} P \Rightarrow f \vee x \\ x \Rightarrow \mathbf{X}(f \vee x) \\ \mathbf{bound} \Rightarrow f \vee \neg x \end{array} \right\} \cup \Psi$$

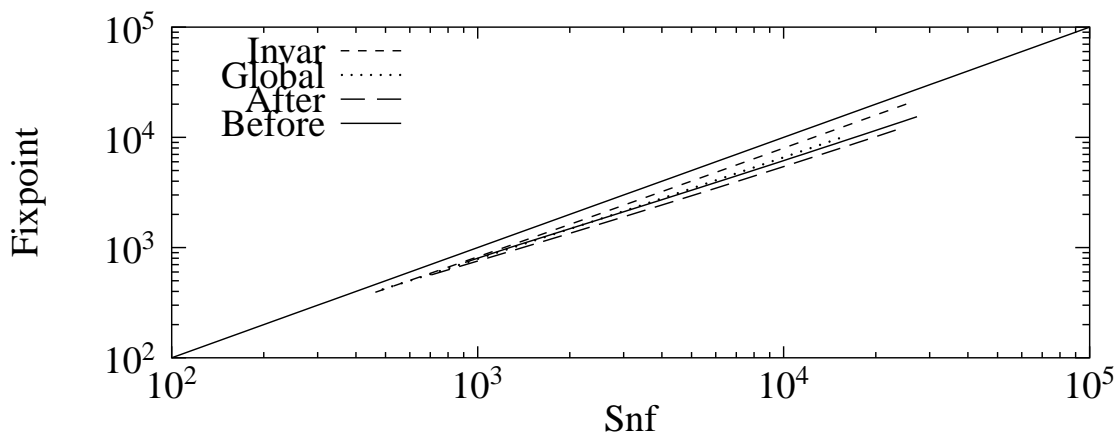
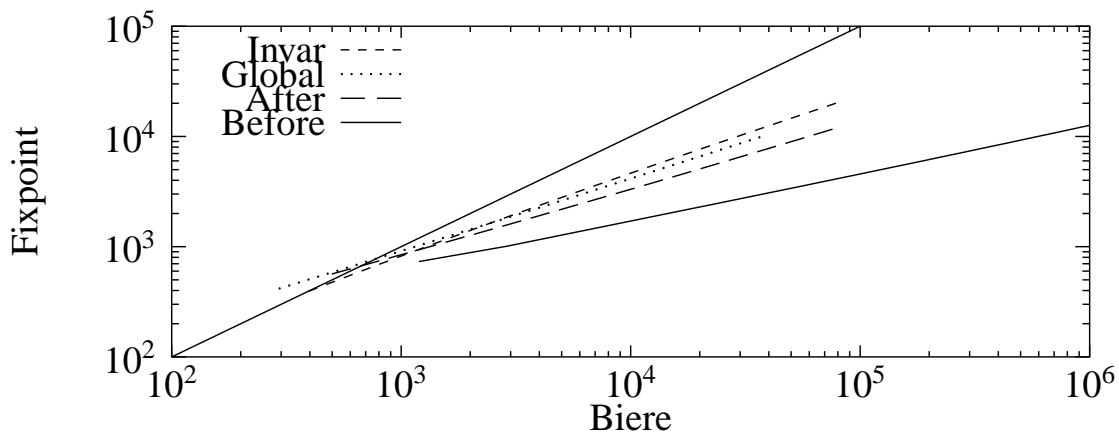
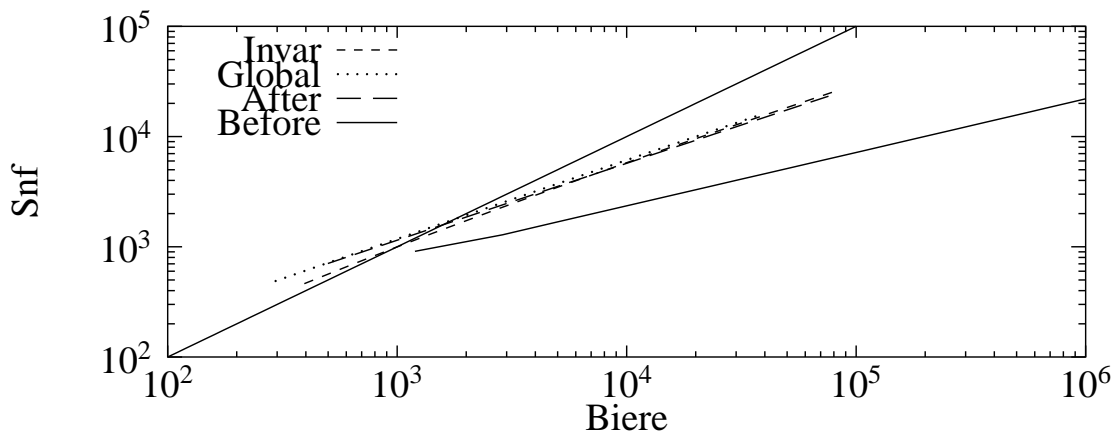
so that only the *next time* operator is used. Each rule now connects two successive states together.

The **bound** operator

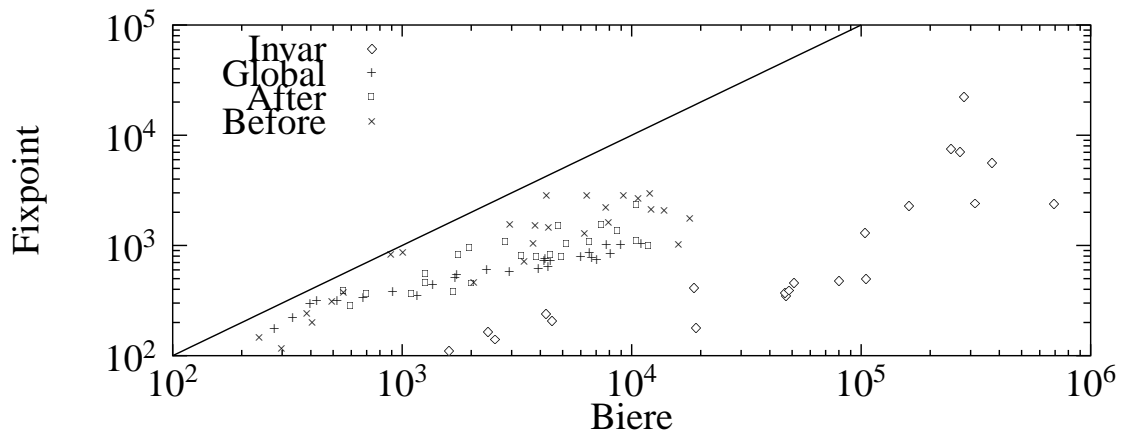
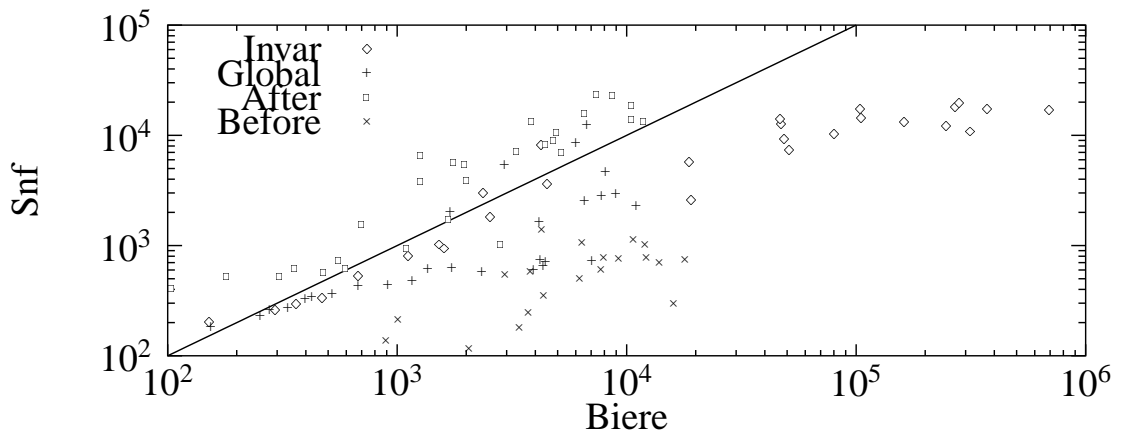
- In Bounded Model Checking we deal with finite representations of infinite paths: loops.
- The rules representing the **F** operator say “either it happens now or in the future”
- A ring of these rules in a loop can evaluate to true, even though the event never occurs!
- The **bound** operator breaks the loop by saying “if we haven’t found the event yet, assume it doesn’t occur”.
- This is correct because we are implementing the *bounded* semantics of LTL.

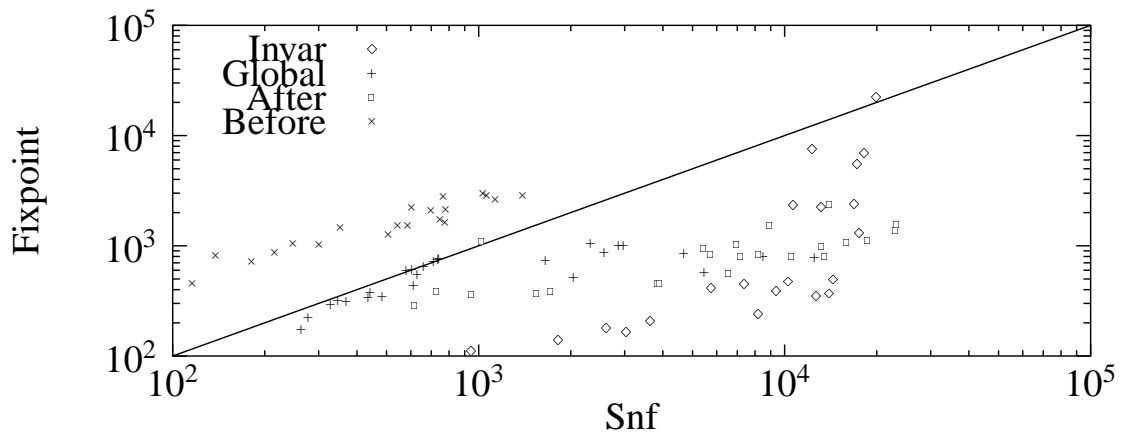
Results (clauses)

Comparing BMC from Biere et al. 1999 with SNF and Fixpoint encodings. Example is a shift register with various specifications.



Results (zChaff decisions))





Conclusions

- Converting specifications to their fixpoint representation always reduces the number of clauses due to increased sharing.
- It also makes the problem easier to solve.
- Converting the *eventually* operator results in improvements over SNF alone.
- Greatest improvements are seen with most complicated specifications

Future directions:

- Better conversion of rules to SAT: don't encode rule instances that won't be needed
- Merging rules: where do rules mean the same thing, so they can be merged together?