

A Fixpoint Encoding for Bounded Model Checking

Daniel Sheridan

dan.sheridan@contact.org.uk



Talk Outline

- Model Checking with SAT
- How Fixpoints fit in
 - Separated Normal Form
 - Fixpoint Normal Form
 - Comparisons
- Some results
- Conclusions

Introduction to SAT

- For example:

$$a \vee b \vee c$$

$$a \vee \neg d$$

$$a \vee d$$

$$b \vee \neg c \vee \neg d$$

$$\neg b \vee d$$

$$\neg c$$

- Find assignments to a , b , c and d to make all of the clauses true
- The quintessential NP-complete problem
- Modern solvers: Grasp (Silva, 1999) Sato (Zhang, 2000) Chaff (Moskewicz, 2001)

Why use SAT?

- State explosion problem
 - Some problems (*eg* multipliers) not efficiently representable in BDDs

Why use SAT?

- State explosion problem
 - Some problems (*eg* multipliers) not efficiently representable in BDDs
- Powerful off-the-shelf solvers with standard file format
- New SAT technology can be plugged in

LTL Model Checking

- Semantics of LTL are over paths π in a Kripke structure M , with path offset i :

$$(M, \pi) \models^i \mathbf{X} f_1 \quad \Leftrightarrow (M, \pi) \models^{i+1} f_1$$

$$(M, \pi) \models^i \mathbf{F} f_1 \quad \Leftrightarrow \exists j, i \leq j. (M, \pi) \models^j f_1$$

$$(M, \pi) \models^i \mathbf{G} f_1 \quad \Leftrightarrow \forall j, i \leq j. (M, \pi) \models^j f_1$$

$$(M, \pi) \models^i [f_1 \mathbf{U} f_2] \Leftrightarrow \exists j, i \leq j. (M, \pi) \models^j f_2$$

$$\wedge \forall n, i \leq n < j. (M, \pi) \models^n f_1$$

- f is a valid LTL formula if it holds for all paths:
 $\forall \pi, (M, \pi) \models^0 f$

Unbounded to Bounded

(Biere et al. 1999)

- Consider a path prefix of length k
 - If $\pi = ab^\omega$ it is a *loop* path, with $\pi(k) = \pi(l)$.
 - Otherwise it is a non-loop path: $\forall l, \pi(k) \neq \pi(l)$
- Semantics redefined to preserve soundness.
- Atomic propositions become vectors of variables:
 x in the model becomes $x[0] \dots x[k]$ for states $\pi(0) \dots \pi(k)$.
- Search for a counterexample:

$$\neg \exists \pi, (M, \pi) \models_k^0 \neg f$$

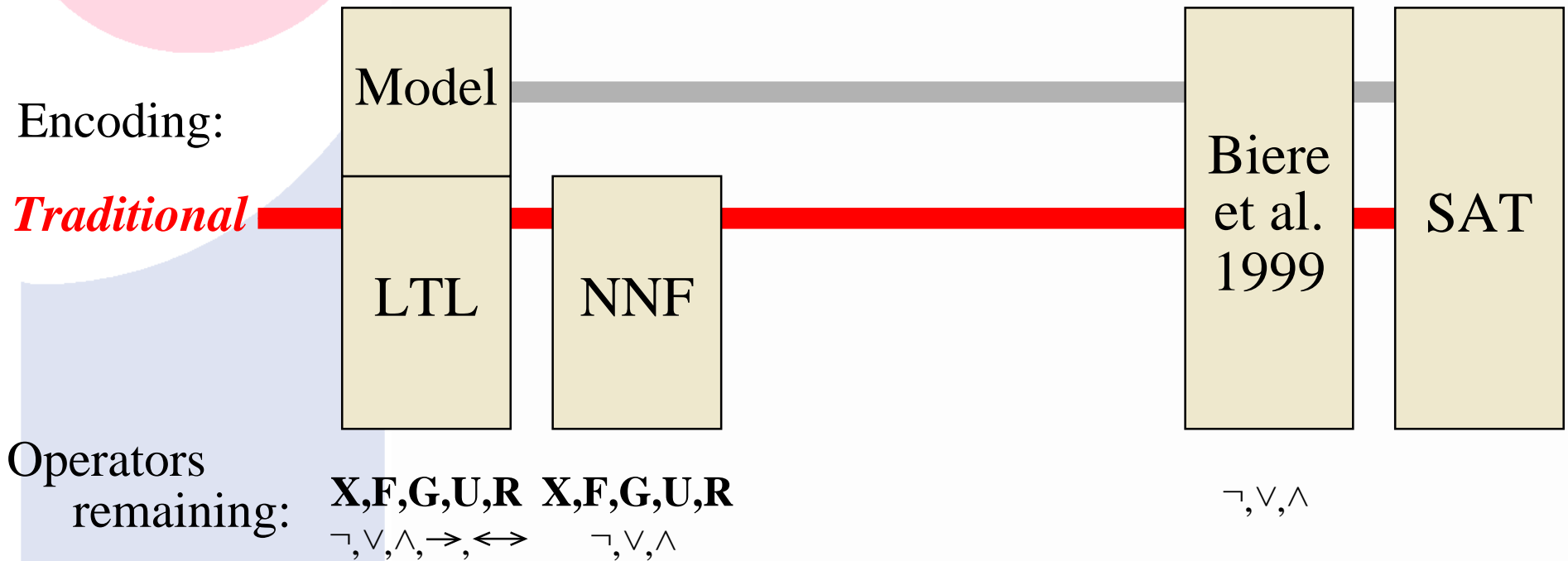
Unbounded to Bounded (2)

- General formula:

$$\llbracket M, f \rrbracket_k := \llbracket M, \pi \rrbracket_k \wedge \left(\neg L_k(\pi) \wedge \llbracket f, \pi \rrbracket_k^0 \vee \bigvee_{l=0}^{k-1} ({}_l L_k(\pi) \wedge {}_l \llbracket f, \pi \rrbracket_k^0) \right)$$

- Loop condition ${}_l L_k \triangleq \pi(l) = \pi(k)$; loop case
encoding of $(M, \pi) \models_k^0 f$ is ${}_l \llbracket f, \pi \rrbracket_k^0$
- Non-loop condition $L_k \triangleq \forall l, {}_l L_k$; non-loop case
encoding of $(M, \pi) \models_k^0 f$ is $\llbracket f, \pi \rrbracket_k^0$
- Biere et al. shows
 $\llbracket M, f \rrbracket_k$ is satisfiable iff $\exists \pi, (M, \pi) \models_k f$

Roadmap



Example: trad. encoding

Is \mathbf{G} (*in* \rightarrow \mathbf{F} *out*) valid for π of length k ?

Example: trad. encoding

Is $\mathbf{G} (in \rightarrow \mathbf{F} out)$ valid for π of length k ?

Negate specification

$$\begin{array}{c} \mathbf{G} (in \rightarrow \mathbf{F} out) \\ \downarrow \\ \neg \mathbf{G} (in \rightarrow \mathbf{F} out) \end{array}$$

Example: trad. encoding

Is $\mathbf{G} (in \rightarrow \mathbf{F} out)$ valid for π of length k ?

Convert to negation normal form

$$\neg \mathbf{G} (in \rightarrow \mathbf{F} out)$$

↓

$$\mathbf{F} (in \wedge \mathbf{G} \neg out)$$

Example: trad. encoding

Is $\mathbf{G} (in \rightarrow \mathbf{F} out)$ valid for π of length k ?

Concentrate on the loop case:

Encode for all loopbacks

$$\mathbf{F} (in \wedge \mathbf{G} \neg out)$$



$$\bigvee_{l=0}^{k-1} ({}_lL_k \wedge {}_l\llbracket \mathbf{F} (in \wedge \mathbf{G} \neg out) \rrbracket_k^0)$$

Example: trad. encoding

Is $\mathbf{G} (in \rightarrow \mathbf{F} out)$ valid for π of length k ?

Concentrate on the loop case:

\mathbf{F} operator becomes disjunction

$$\bigvee_{l=0}^{k-1} ({}_l L_k \wedge {}_l \llbracket \mathbf{F} (in \wedge \mathbf{G} \neg out) \rrbracket_k^0)$$

↓

$$\bigvee_{l=0}^{k-1} ({}_l L_k \wedge \bigvee_{i=0}^k {}_l \llbracket in \wedge \mathbf{G} \neg out \rrbracket_k^i)$$

Example: trad. encoding

Is $\mathbf{G} (in \rightarrow \mathbf{F} out)$ valid for π of length k ?

Concentrate on the loop case:

Propositional connectives are trivial

$$\bigvee_{l=0}^{k-1} (lL_k \wedge \bigvee_{i=0}^k l \llbracket in \wedge \mathbf{G} \neg out \rrbracket_k^i)$$

↓

$$\bigvee_{l=0}^{k-1} (lL_k \wedge \bigvee_{i=0}^k (in[i] \wedge l \llbracket \mathbf{G} \neg out \rrbracket_k^i))$$

Example: trad. encoding

Is $\mathbf{G}(in \rightarrow \mathbf{F} out)$ valid for π of length k ?

Concentrate on the loop case:

\mathbf{G} operator becomes conjunction

$$\bigvee_{l=0}^{k-1} ({}_l L_k \wedge \bigvee_{i=0}^k (in[i] \wedge {}_l \llbracket \mathbf{G} \neg out \rrbracket_k^i))$$

↓

$$\bigvee_{l=0}^{k-1} ({}_l L_k \wedge \bigvee_{i=0}^k (in[i] \wedge \bigwedge_{j=\min(i,l)}^k \neg out[j]))$$

Observations

- Encoding depends on loop existence and location
- Gets very complicated with nested operators
- Space and time complexity $O(k^n)$ where n is number of temporal operators
- Biere et al. claim $O(k^2)$ — assuming ideal structure sharing

Talk Outline

- Model Checking with SAT
- How Fixpoints fit in
 - Separated Normal Form
 - Fixpoint Normal Form
 - Comparisons
- Some results
- Conclusions

Fixpoints - recap

- Greatest fixpoint: the largest Z such that $\tau(Z) = Z$. Written $\nu Z.\tau(Z)$.
- Least fixpoint: the smallest Z such that $\tau(Z) = Z$. Written $\mu Z.\tau(Z)$.
- Tarski's fixpoint theorem (1955):
 - Start with $Z = \perp$ or $Z = \top$
 - Let $Z \leftarrow \tau(Z)$
 - Repeat until $\tau(Z) = Z$

Fixpoints in LTL

- Can characterise LTL operators by fixpoint formulæ
- Partial order: *sets of paths* under *subset inclusion*
- Emerson and Clarke (1980):
 - $\mathbf{G} f = \nu Z.f \wedge \mathbf{X} Z$
 - $\mathbf{F} f = \mu Z.f \vee \mathbf{X} Z$
 - $f_1 \mathbf{U} f_2 = \mu Z.f_2 \vee (f_1 \wedge \mathbf{X} Z)$
- Basis of Symbolic Model Checking (McMillan, 1993)

Separated Normal Form

- Gabbay's Separation Theorem (1989):
Any temporal formula may be converted to the form

$$\mathbf{G} \bigwedge_i (P_i \rightarrow F_i)$$

Past time formulæ P_i , future time formulæ F_i

Separated Normal Form

- Gabbay's Separation Theorem (1989):
 Any temporal formula may be converted to the form

$$\mathbf{G} \bigwedge_i (P_i \rightarrow F_i)$$

Past time formulæ P_i , future time formulæ F_i

- LTL doesn't have any past: introduce **start** and **bound**:
 - $(M, f) \models_k^i \mathbf{start} \Leftrightarrow i = 0$
 - $(M, f) \models_k^i \mathbf{bound} \Leftrightarrow i = k$

Separated Normal Form

- Fisher (1991): rules in SNF

start $\Rightarrow \bigvee_j l_j$ An *initial* rule

$\bigwedge_i l_i \Rightarrow \mathbf{X} \bigvee_j l_j$ A *global X*-rule

$\bigwedge_i l_i \Rightarrow \mathbf{F} \bigvee_j l_j$ A *global F*-rule

- l_i, l_j are variables; indicate rules with \Rightarrow operator
- Formula becomes a set of rules (**G** is implied)
- Greatest fixpoint only

Separated Normal Form

- Transformations

- From $\mathbf{G} f = \nu Z.f \wedge \mathbf{X} Z$:

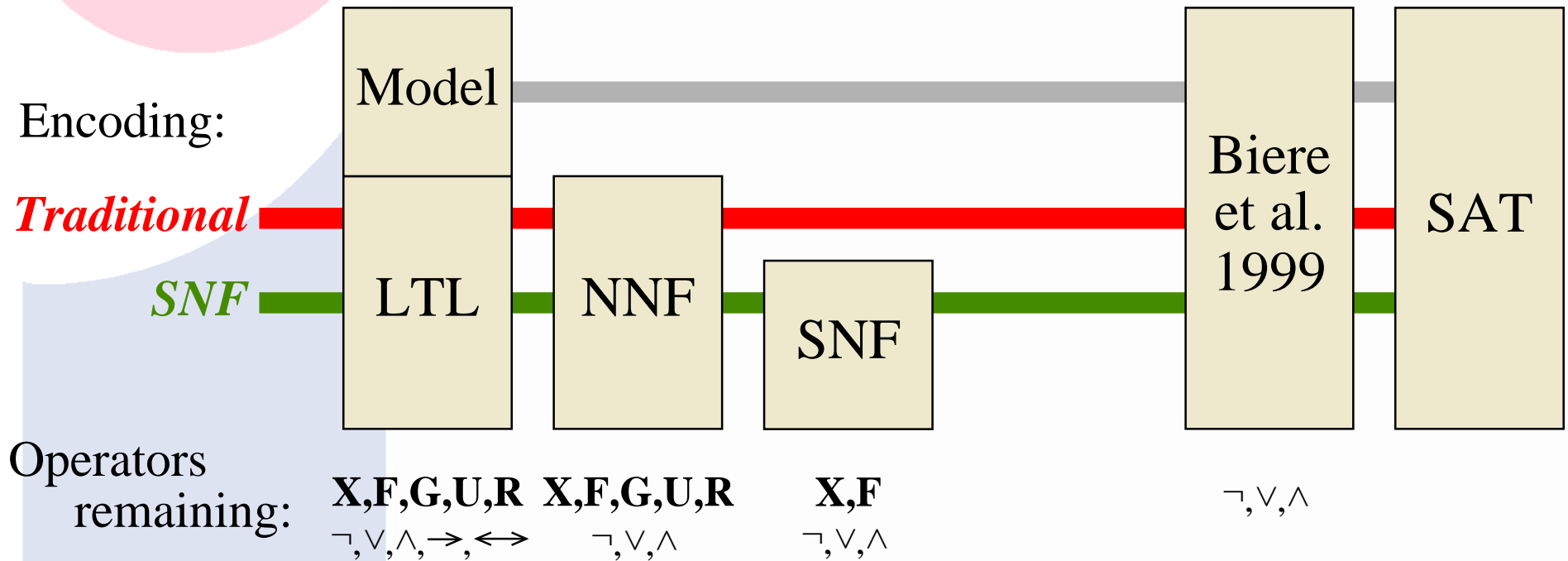
$$P \Rightarrow \mathbf{G} f \longrightarrow \begin{cases} P \Rightarrow f \wedge x \\ x \Rightarrow \mathbf{X}(f \wedge x) \end{cases}$$

- Renaming subformulae:

$$P \Rightarrow \mathcal{F}(f_1) \longrightarrow \begin{cases} P \Rightarrow \mathcal{F}(x) \\ x \Rightarrow f_1 \end{cases}$$

- New variables: implicitly existential

Roadmap



Example: SNF encoding

Is $\mathbf{G} (in \rightarrow \mathbf{F} out)$ valid for π of length k ?

Convert to negation normal form

$$\mathbf{F} (in \wedge \mathbf{G} \neg out)$$

Example: SNF encoding

Is $\mathbf{G} (in \rightarrow \mathbf{F} out)$ valid for π of length k ?

Convert to a set of rules

$$\{\mathbf{start} \Rightarrow \mathbf{F} (in \wedge \mathbf{G} \neg out)\}$$

Example: SNF encoding

Is $\mathbf{G} (in \rightarrow \mathbf{F} out)$ valid for π of length k ?

Subformula renaming

$$\left\{ \mathbf{start} \Rightarrow \mathbf{F} (in \wedge \mathbf{G} \neg out) \right\} \longrightarrow \left\{ \begin{array}{l} \mathbf{start} \Rightarrow \mathbf{F} x_0 \\ x_0 \Rightarrow (in \wedge \mathbf{G} \neg out) \end{array} \right\}$$

Example: SNF encoding

Is $\mathbf{G} (in \rightarrow \mathbf{F} out)$ valid for π of length k ?

Common past transformation

$$\left\{ \begin{array}{l} \mathbf{start} \Rightarrow \mathbf{F} x_0 \\ x_0 \Rightarrow (in \wedge \mathbf{G} \neg out) \end{array} \right\} \longrightarrow \left\{ \begin{array}{l} \mathbf{start} \Rightarrow \mathbf{F} x_0 \\ x_0 \Rightarrow in \\ x_0 \Rightarrow \mathbf{G} \neg out \end{array} \right\}$$

Example: SNF encoding

Is $\mathbf{G} (in \rightarrow \mathbf{F} out)$ valid for π of length k ?

Fixpoint characterisation of \mathbf{G}

$$\left\{ \begin{array}{l} \mathbf{start} \Rightarrow \mathbf{F} x_0 \\ x_0 \Rightarrow in \\ x_0 \Rightarrow \mathbf{G} \neg out \end{array} \right\} \longrightarrow \left\{ \begin{array}{l} \mathbf{start} \Rightarrow \mathbf{F} x_0 \\ x_0 \Rightarrow in \\ x_0 \Rightarrow \neg out \wedge x_1 \\ x_1 \Rightarrow \mathbf{X} (\neg out \wedge x_1) \end{array} \right\}$$

Example: SNF encoding

Is $\mathbf{G} (in \rightarrow \mathbf{F} out)$ valid for π of length k ?

Common past transformation

$$\left\{ \begin{array}{l} \mathbf{start} \Rightarrow \mathbf{F} x_0 \\ x_0 \Rightarrow in \\ x_0 \Rightarrow \neg out \wedge x_1 \\ x_1 \Rightarrow \mathbf{X} (\neg out \wedge x_1) \end{array} \right\} \longrightarrow \left\{ \begin{array}{l} \mathbf{start} \Rightarrow \mathbf{F} x_0 \\ x_0 \Rightarrow in \\ x_0 \Rightarrow \neg out \\ x_0 \Rightarrow x_1 \\ x_1 \Rightarrow \mathbf{X} \neg out \\ x_1 \Rightarrow \mathbf{X} x_1 \end{array} \right\}$$

Example: SNF encoding

Is $\mathbf{G} (in \rightarrow \mathbf{F} out)$ valid for π of length k ?

Encode to propositional logic (with \mathbf{G})

$$\left\{ \begin{array}{l}
 \mathbf{start} \Rightarrow \mathbf{F} x_0 \\
 x_0 \Rightarrow in \\
 x_0 \Rightarrow \neg out \\
 x_0 \Rightarrow x_1 \\
 x_1 \Rightarrow \mathbf{X} \neg out \\
 x_1 \Rightarrow \mathbf{X} x_1
 \end{array} \right\} \longrightarrow \begin{array}{l}
 \bigvee_{i=0}^k x_0[i] \\
 \wedge \bigwedge_{i=0}^k x_0[i] \rightarrow in[i] \\
 \wedge \bigwedge_{i=0}^k x_0[i] \rightarrow \neg out[i] \\
 \wedge \bigwedge_{i=0}^k x_0[i] \rightarrow x_1[i] \\
 \wedge \bigwedge_{i=0}^{k-1} x_1[i] \rightarrow \neg out[i+1] \\
 \wedge \bigwedge_{i=0}^{k-1} x_1[i] \rightarrow x_1[i+1]
 \end{array}$$

Comparison

$\neg \mathbf{G} (in \rightarrow \mathbf{F} out)$ in the loop case

- Traditional encoding:

$$\bigvee_{l=0}^{k-1} ({}_l L_k \wedge \bigvee_{i=0}^k (in[i] \wedge \bigwedge_{j=\min(i,l)}^k \neg out[i]))$$

- SNF (trad encoding for F):

$$\begin{aligned} & \bigvee_{l=0}^{k-1} {}_l L_k \wedge \bigvee_{i=0}^k x_0[i] \wedge \\ & \bigwedge_{i=0}^k (x_0[i] \rightarrow x_1[i] \wedge x_0[i] \rightarrow in[i] \wedge x_0[i] \rightarrow \neg out[i]) \\ & \bigwedge_{i=0}^{k-1} (x_1[i] \rightarrow \neg out[i+1] \wedge x_1[i] \rightarrow x_1[i+1]) \end{aligned}$$

Separated Normal Form

- Transformation produces 1 or 2 rules for each operator
- Encodes 1, k , or k^2 clauses for each rule
- $O(nk^2)$ in time and space!

Separated Normal Form

- Transformation produces 1 or 2 rules for each operator
- Encodes 1, k , or k^2 clauses for each rule
- $O(nk^2)$ in time and space!
- Worst case is **F** operator: encodes to k^2 symbols
- SNF retains **F** operator because of unbounded time; we are dealing with *bounded* time!
- Conjecture: We benefit when encoding to SAT if we unfold **F** too

Talk Outline

- Model Checking with SAT
- How Fixpoints fit in
 - Separated Normal Form
 - Fixpoint Normal Form
 - Comparisons
- Some results
- Conclusions

Fixpoint Form

start $\Rightarrow \bigvee_j l_j$ An *initial* rule

bound $\Rightarrow \bigvee_j l_j$ A *final* rule

$\bigwedge_i l_i \Rightarrow \mathbf{X} \bigvee_j l_j$ A *global X*-rule

- Each rule looks 0 or 1 steps ahead

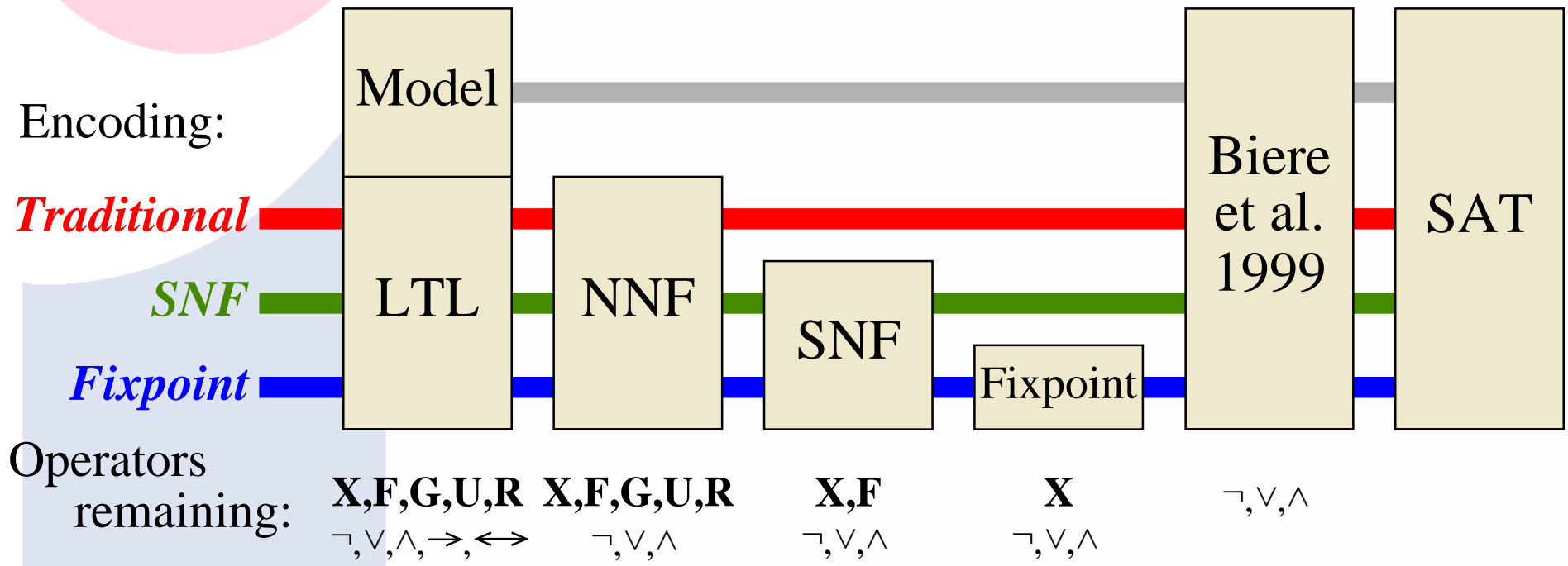
Fixpoint Form

- New transformation from $\mathbf{F} f = \mu Z.f \vee \mathbf{X} Z$:

$$P \Rightarrow \mathbf{F} f \longrightarrow \begin{cases} P \Rightarrow f \vee x \\ x \Rightarrow \mathbf{X} (f \vee x) \\ \mathbf{bound} \Rightarrow \neg x \end{cases}$$

- Use of **bound** “pins down” fixpoint so greatest fixpoint (from \exists) = least fixpoint
- For *SNF encoding* we used the traditional encoding of \mathbf{F} ; for *Fixpoint encoding* we use this transformation

Roadmap



Example: Fixpoint encoding

Is $\mathbf{G} (in \rightarrow \mathbf{F} out)$ valid for π of length k ?

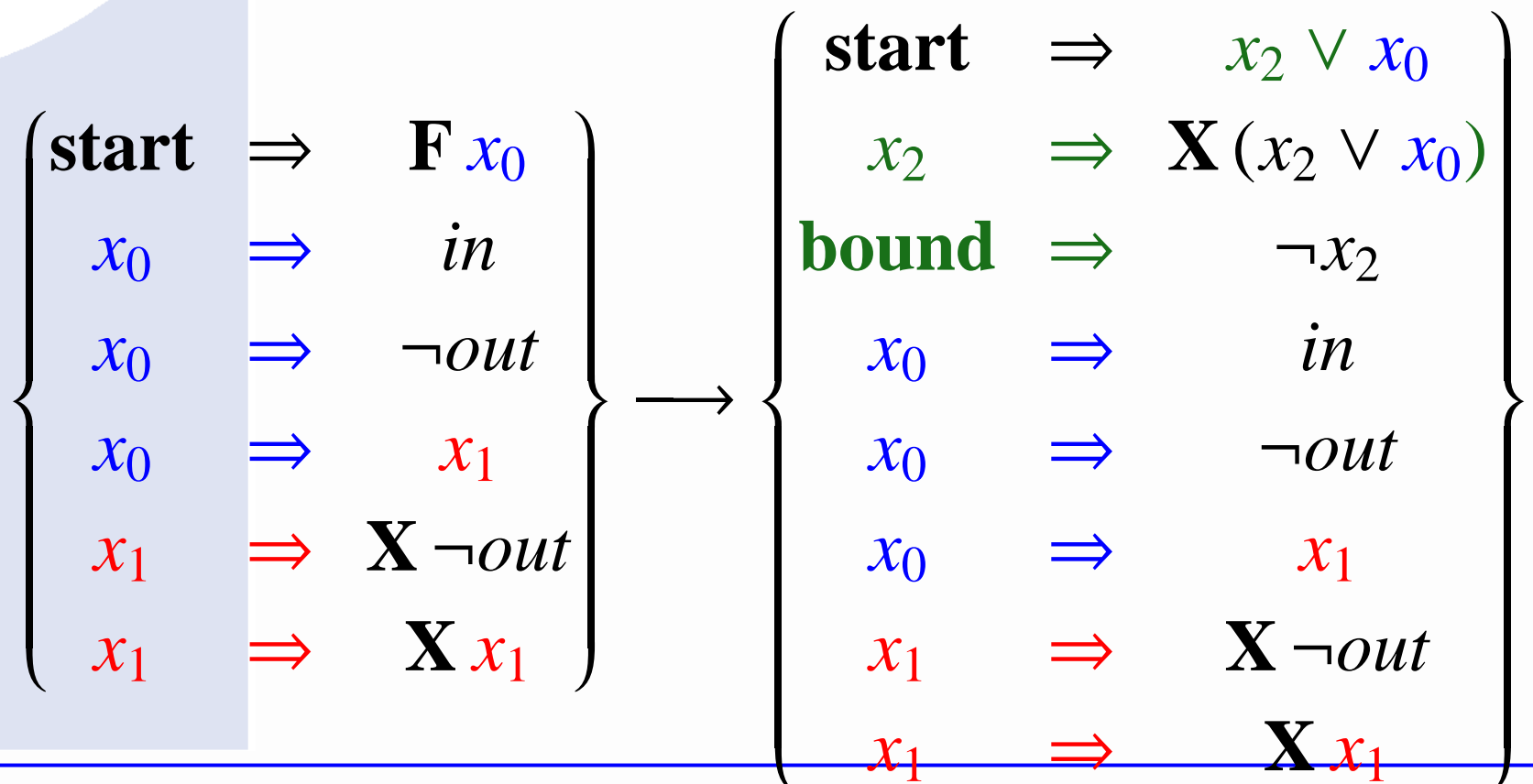
Convert to SNF

$$\left\{ \begin{array}{l} \mathbf{start} \Rightarrow \mathbf{F} x_0 \\ x_0 \Rightarrow in \\ x_0 \Rightarrow \neg out \\ x_0 \Rightarrow x_1 \\ x_1 \Rightarrow \mathbf{X} \neg out \\ x_1 \Rightarrow \mathbf{X} x_1 \end{array} \right\}$$

Example: Fixpoint encoding

Is $\mathbf{G} (in \rightarrow \mathbf{F} out)$ valid for π of length k ?

Fixpoint characterisation of \mathbf{F}



Example: Fixpoint encoding

Is $\mathbf{G} (in \rightarrow \mathbf{F} out)$ valid for π of length k ?

Encode to propositional logic

$$\left\{ \begin{array}{l}
 \mathbf{start} \Rightarrow x_2 \vee x_0 \\
 x_2 \Rightarrow \mathbf{X} (x_2 \vee x_0) \\
 \mathbf{bound} \Rightarrow \neg x_2 \\
 x_0 \Rightarrow in \\
 x_0 \Rightarrow \neg out \\
 \vdots
 \end{array} \right\} \longrightarrow \bigwedge_{i=0}^{k-1} \left(\begin{array}{l}
 x_2[0] \vee x_0[0] \\
 x_2[i] \rightarrow (x_2[i+1] \vee x_0[i+1]) \\
 \neg x_2[k] \\
 x_0[i] \rightarrow in[i] \\
 x_0[i] \rightarrow \neg out[i] \\
 \vdots
 \end{array} \right)$$

Comparison: $\neg G$ (*in* \rightarrow *F out*)

- Traditional encoding:

$$\bigvee_{l=0}^{k-1} ({}_lL_k \wedge \bigvee_{i=0}^k (in[i] \wedge \bigwedge_{j=\min(i,l)}^k \neg out[i]))$$

- Fixpoint (transformation for F):

$$\begin{aligned} & \bigvee_{l=0}^{k-1} {}_lL_k \wedge (x_2[0] \vee x_0[0]) \wedge \neg x_2[k] \wedge \\ & \bigwedge_{i=0}^k (x_0[i] \rightarrow \neg out[i] \wedge x_0[i] \rightarrow x_1[i] \wedge x_0[i] \rightarrow in[i]) \\ & \bigwedge_{i=0}^{k-1} (x_2[i] \rightarrow (x_2[i+1] \vee x_0[i+1]) \wedge x_1[i] \rightarrow \neg out[i+1] \\ & \quad \wedge x_1[i] \rightarrow x_1[i+1]) \end{aligned}$$

Talk Outline

- Model Checking with SAT
- How Fixpoints fit in
 - Separated Normal Form
 - Fixpoint Normal Form
 - Comparisons
- Some results
- Conclusions

Advantages of SNF/Fixpoint

- Rules have a known form
 - Each can be encoded optimally
 - Can identify subformulae common to non-loop and loop cases
 - Sometimes eliminate the loop conditions

Advantages of SNF/Fixpoint

- Rules have a known form
 - Each can be encoded optimally
 - Can identify subformulae common to non-loop and loop cases
 - Sometimes eliminate the loop conditions
- Encoding is linear or quadratic in time and space
 - n is number of temporal operators
 - Conversion to SNF/Fixpoint is $O(n)$
 - Encoding SNF is $O(nk^2)$
 - Encoding Fixpoint is $O(kn)$; traditional can be $O(k^n)$

Talk Outline

- Model Checking with SAT
- How Fixpoints fit in
 - Separated Normal Form
 - Fixpoint Normal Form
 - Comparisons
- **Some results**
- Conclusions

Implementation

- NuSMV 2 (a GPLed model checker — Cimatti et al. 1999) includes a bounded model checker
 - SNF and Fixpoint encodings implemented as additional encoding for specification
 - Only additional change is to the CNF conversion
- Expected to be included in the next public release
- Try it on your problems soon!

Results overview

- Results with zChaff SAT solver
- Compare NuSMV encoding (as Biere et al 1999) against SNF and Fixpoint encodings
- Examine:
 - “Toy” problems: Shift register, DME
 - Some of Texas-97 benchmark suite

Results: Texas '97

MSI cache coherence protocol:

Specification	k	Trad	SNF	Fixpoint
$\mathbf{G}(bus_reqA \rightarrow \mathbf{X} bus_ackA)$	20	39.22	8.2	5.79
$\mathbf{G}(bus_reqB \rightarrow \mathbf{F} bus_ackB)$	20	54.94	62.11	40.25
$\mathbf{G}(bus_reqC \rightarrow \mathbf{F} bus_ackC)$	20	44.8	50.27	37.65

Instruction fetch control module:

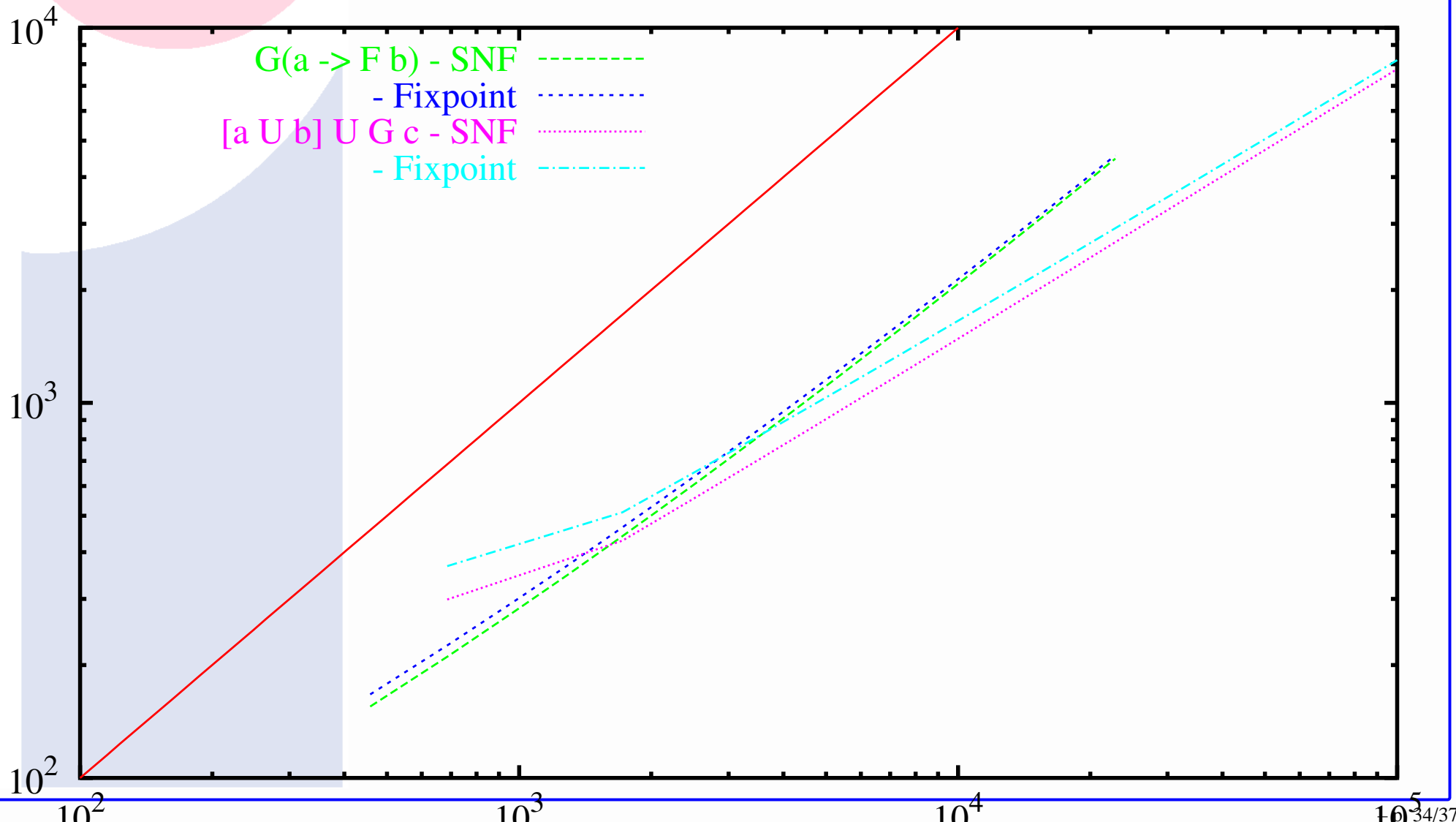
Specification	k	Trad	SNF	Fixpoint
$\mathbf{G}(IStall_1 \rightarrow \mathbf{X} IStall_2)$	10	1.29	0.39	0.50
$\mathbf{G}((Pres_1 = R) \rightarrow \mathbf{X}((Prev_2 = R)))$	10	3.74	1.49	1.88
$\neg[\neg Tag_2 \mathbf{U} (Cache_2 \wedge \neg Tag_2)]$	10	2.78	2.24	1.40

Results: DME

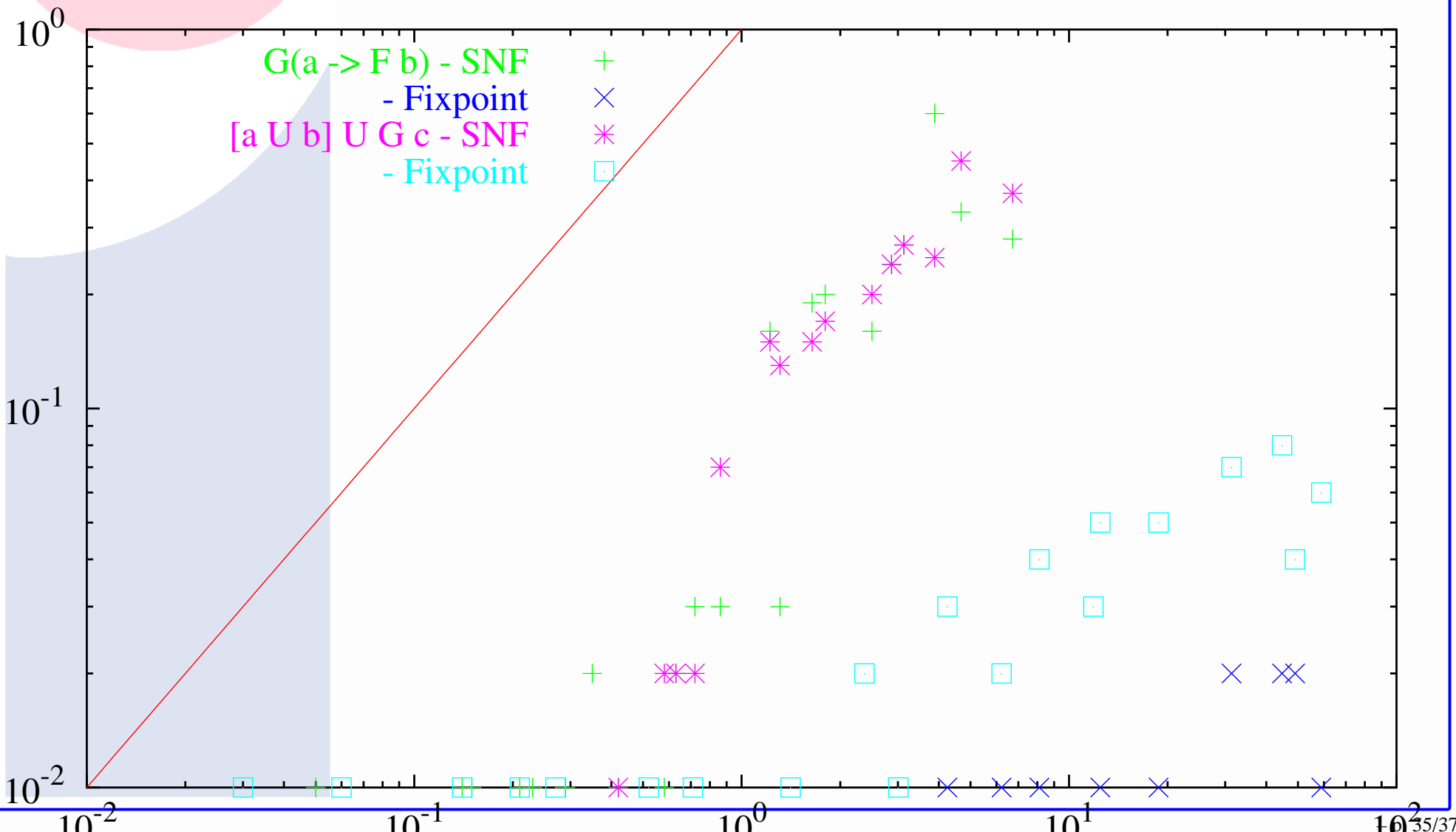
Specification	Depth	k	Trad	SNF	Fixpoint
Accessibility	2	30	2.65	0.33	0.36
Accessibility	2	40	20.93	4.84	4.33
Priority for 0	2	14	0.13	0.02	0.02
Priority for 1	2	54	14.93	0.44	0.76
Overtaking by 1	4	40	85.73	2.15	1.11
Overtaking by 2	6	40	*	4.92	5.15

* encoding time exceeded 30 minutes

Trad vs SNF/Fixpoint (Size)



Trad vs SNF/Fixpoint (Time)



Talk Outline

- Model Checking with SAT
- How Fixpoints fit in
 - Separated Normal Form
 - Fixpoint Normal Form
 - Comparisons
- Some results
- **Conclusions**

Conclusions

- The SNF encoding usually gives an improvement in the time taken by the SAT solver over the traditional encoding
 - Performance depends on the specification
- The Fixpoint encoding can result in faster solving times than the SNF encoding
 - Performance depends on the specification
- The SNF encoding and Fixpoint encoding are faster to compute than the traditional encoding