Memory Model-aware Testing

a Unified Complexity Analysis

Florian Furbach, Roland Meyer,
Klaus Schneider, and Maximilian Senftleben

University of Kaiserslautern
Department of Computer Science
Germany

ACSD 2014

June 26, 2014

- Programmers expect sequential consistency.

Gibbons, Korach 1997
Cantin, Lipasti, Smith 2005

Introduction

Motivation

- Programmers expect sequential consistency.
- Modern architectures lack sequential consistency.

Gibbons, Korach 1997
Cantin, Lipasti, Smith 2005

Introduction

Motivation

- Programmers expect sequential consistency.
- Modern architectures lack sequential consistency.
- Modern architectures employ weak memory models.

Gibbons, Korach 1997
Cantin, Lipasti, Smith 2005

Introduction

Motivation

- Programmers expect sequential consistency.
- Modern architectures lack sequential consistency.
- Modern architectures employ weak memory models.
- Weak memory models may introduce undesired states.

Gibbons, Korach 1997
Cantin, Lipasti, Smith 2005

Introduction
Motivation

- Programmers expect sequential consistency.
- Modern architectures lack sequential consistency.
- Modern architectures employ weak memory models.
- Weak memory models may introduce undesired states.
- State explosion for reachability analysis.

Gibbons, Korach 1997
Cantin, Lipasti, Smith 2005

Introduction

Motivation

- Programmers expect sequential consistency.
- Modern architectures lack sequential consistency.
- Modern architectures employ weak memory models.
- Weak memory models may introduce undesired states.
- State explosion for reachability analysis.

- **Complexity of Testing?**

Gibbons, Korach 1997
Cantin, Lipasti, Smith 2005

- Test: sequences of reads/writes for multiple processes.
- Reads are blocking.
- Memory variables initialized to 0.

EXAMPLE: Test $\mathcal{T}$

P1
(w, x, 1)
(w, x, 2)

P2
(r, x, 1)

Introduction

Notions

- Test: sequences of reads/writes for multiple processes.
- Reads are blocking.
- Memory variables initialized to 0.

EXAMPLE: Test $\mathcal{T}$

P1
(w, x, 1)
(w, x, 2)

P2
(r, x, 1)

◀ :

x : 0

- Test: sequences of reads/writes for multiple processes.
- Reads are blocking.
- Memory variables initialized to 0.

EXAMPLE: Test $\mathcal{T}$

P1

(w, x, 1)
(w, x, 2)

P2

(r, x, 1)

$$\blacktriangleleft : \quad (w, x, 1)$$
$$x : \quad 1$$

Introduction

Notions

- Test: sequences of reads/writes for multiple processes.
- Reads are blocking.
- Memory variables initialized to 0.

EXAMPLE: Test $\mathcal{T}$

P1
(w, x, 1)
(w, x, 2)

P2
(r, x, 1)

$$\blacktriangleleft : \quad (w, x, 1).(r, x, 1)$$
$$x : \quad 1$$

- Test: sequences of reads/writes for multiple processes.
- Reads are blocking.
- Memory variables initialized to 0.

EXAMPLE: Test $\mathcal{T}$

P1
(w, x, 1)
(w, x, 2)

P2
(r, x, 1)

$$\blacktriangleleft: \quad (w,x,1).(r,x,1).(w,x,2)$$
$$x: \quad 2$$

Serial View

- Processes observe operations in different orders (views).
- A **serial view** ◀ = $\mathit{SerialView}(\mathcal{O}, <)$ is a sequence of operations from $\mathcal{O}$ that respects some partial order $<$.
- Always read from last write.

Steinke, Nutt 2004

Serial View

- Processes observe operations in different orders (views).
- A **serial view** $\blacktriangleleft = \mathit{SerialView}(\mathcal{O}, <)$ is a sequence of operations from $\mathcal{O}$ that respects some partial order $<$.
- Always read from last write.

- A Test $\mathcal{T}$ is **executable under sequential consistency** if:

$$\exists \quad \blacktriangleleft = \mathit{SerialView}(\mathcal{T}, <_{PO}).$$

Example $\blacktriangleleft$: $\quad (w, x, 1).(r, x, 1).(w, x, 2)$

Steinke, Nutt 2004

The Testing Problem

Testing Problem of model M:

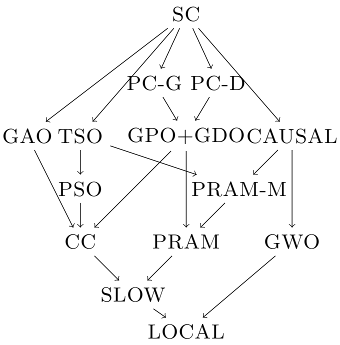
Given test $\mathcal{T}$, is it **executable under** *model* **M***?*

Steinke, Nutt 2004

The Testing Problem

Testing Problem of model M:

Given test $\mathcal{T}$, is it **executable under** *model* **M***?*

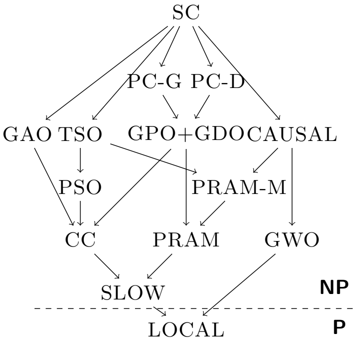


Steinke, Nutt 2004

The Testing Problem

Testing Problem of model M:

Given test $\mathcal{T}$, is it **executable under** *model* **M***?*



Steinke, Nutt 2004

The Testing Problem

- **Testing Problem is in NP for all models**
- Testing Problem is NP-hard for most models
- Testing Problem is in P for some models

Testing is in NP

Uniform Reduction to SAT:

- Formula:

  $$WT(\mathcal{T}) \wedge \mathrm{SV}_1 \wedge .. \wedge \mathrm{SV}_k$$

  WT: Unique Writes-To
  SV: SerialView properties

Testing is in NP

Uniform Reduction to SAT:

- Formula:

  $$WT(\mathcal{T}) \wedge \mathrm{SV}_1 \wedge .. \wedge \mathrm{SV}_k$$

  WT: Unique Writes-To
  SV: SerialView properties

- Boolean variable:

  $$sv_{i,j} \leftrightarrow (op_i \blacktriangleleft op_j)$$

- Serial view properties:

  Totality, Asymmetry, Transitivity, Read-Last-Write

The Testing Problem

- **Testing Problem is in NP for all models**
  - Uniform SAT reduction.
  - Optimal solution if NP-hard.
- Testing Problem is NP-hard for most models
- Testing Problem is in P for some models

The Testing Problem

- Testing Problem is in NP for all models
- **Testing Problem is NP-hard for most models**
- Testing Problem is in P for some models

The Testing Problem

- Testing Problem is in NP for all models
- **Testing Problem is NP-hard for most models**
  - Our proofs cover multiple models
- Testing Problem is in P for some models

NP-hard for most models

Range reduction

$\mathbf{M}_{Strong} \preceq \mathbf{M}_{Weak}$**-range reduction** *$f$ of* SAT *to testing*:

$$
\begin{tikzcd}
 & & & \mathrm{M}_{Strong} \arrow[dl] \arrow[d] \arrow[dr] & \\
\mathrm{SAT} & & \mathrm{M} \arrow[r, dotted, no head] \arrow[dr] & \mathrm{M}' \arrow[r, dotted, no head] \arrow[d] & \mathrm{M}'' \arrow[dl] \\
 & & & \mathrm{M}_{Weak} &
\end{tikzcd}
$$

(i) ϕ is SAT $\Longrightarrow$ test $f(\phi)$ is executable under M_{Strong}.

(ii) test $f(\phi)$ is executable under M_{Weak} $\Longrightarrow$ ϕ is SAT.

NP-hard for most models

Range reduction

$\mathbf{M}_{Strong} \preceq \mathbf{M}_{Weak}$**-range reduction** *$f$ of* SAT *to testing*:

$$\begin{tikzcd}
& & \mathrm{M}_{\mathit{Strong}} \arrow[dl] \arrow[d] \arrow[dr] & \\
\mathrm{SAT} \arrow[urr, Rightarrow, "\text{(i)}"] & \mathrm{M} \arrow[r, dotted, no head] \arrow[dr] & \mathrm{M}' \arrow[r, dotted, no head] \arrow[d] & \mathrm{M}'' \arrow[dl] \\
& & \mathrm{M}_{\mathit{Weak}} &
\end{tikzcd}$$

(i) ϕ is SAT $\Longrightarrow$ test $f(\phi)$ is executable under M_{Strong}.

(ii) test $f(\phi)$ is executable under $\mathrm{M}_{Weak} \Longrightarrow \phi$ is SAT.

NP-hard for most models

Range reduction

$\mathbf{M}_{Strong} \preceq \mathbf{M}_{Weak}$**-range reduction** *$f$ of* SAT *to testing*:

$$\begin{tikzcd}
 & & \mathrm{M}_{Strong} \arrow[dl] \arrow[d] \arrow[dr] & \\
\mathrm{SAT} \arrow[urr, Rightarrow, "\text{(i)}"] & \mathrm{M} \arrow[r, dotted, no head] \arrow[dr] & \mathrm{M}' \arrow[r, dotted, no head] \arrow[d] & \mathrm{M}'' \arrow[dl] \\
 & & \mathrm{M}_{Weak} &
\end{tikzcd}$$

(i) ϕ is SAT $\Longrightarrow$ test $f(\phi)$ is executable under M_{Strong}.

(ii) test $f(\phi)$ is executable under $\mathrm{M}_{Weak} \Longrightarrow \phi$ is SAT.

NP-hard for most models

Range reduction

$\mathbf{M}_{Strong} \preceq \mathbf{M}_{Weak}$-range reduction *$f$ of SAT to testing*:

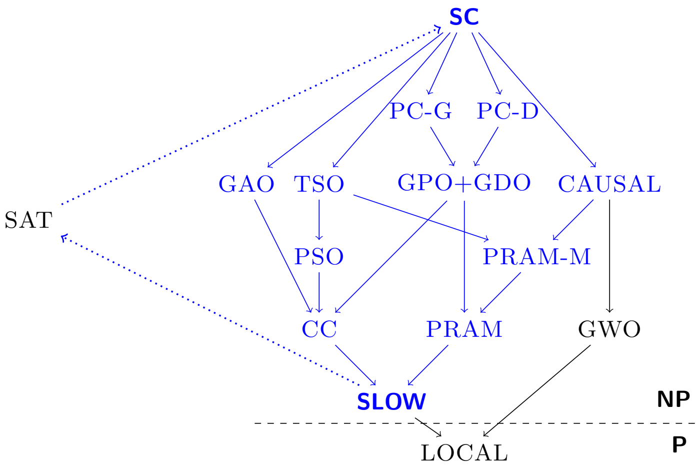
$$
\begin{tikzcd}
 & & \mathrm{M}_{Strong} \arrow[dl] \arrow[d] \arrow[dr] & \\
\mathrm{SAT} \arrow[urr, Rightarrow, "\text{(i)}"] & \mathrm{M} \arrow[dr] & \mathrm{M}' \arrow[d] & \mathrm{M}'' \arrow[dl] \\
 & & \mathrm{M}_{Weak} \arrow[ull, Rightarrow, "\text{(ii)}"] &
\end{tikzcd}
$$

(i) ϕ is SAT $\implies$ test $f(\phi)$ is executable under M_{Strong}.

(ii) test $f(\phi)$ is executable under $\mathrm{M}_{Weak} \implies \phi$ is SAT.

NP-hard for most models

SC ≤ SLOW-Range-Reduction



NP-hard for most models

Slow Consistency

Writes from one process to one variable are observed in same order by all processes.

Hutto, Ahamad 1990

NP-hard for most models

Slow Consistency

Writes from one process to one variable are observed in same order by all processes.

- The program order is respected.
- For each process p and variable x: there exists a serial view on all writes to x and reads from x of p.

$$\forall x, p\ \exists \quad \blacktriangleleft = SerialView(\quad \mathcal{T}|_{w,x} \cup \mathcal{T}|_{p,x} \quad , <_{PO})$$

$$\exists \quad \blacktriangleleft = SerialView(\quad \mathcal{T} \quad , \qquad\qquad <_{PO}) \qquad\qquad [SC]$$

Hutto, Ahamad 1990

NP-hard for most models

Slow Consistency

Writes from one process to one variable are observed in same order by all processes.

- The program order is respected.
- For each process p and variable x: there exists a serial view on all writes to x and reads from x of p.

$$\forall x, p \; \exists \; \blacktriangleleft = \mathit{SerialView}(\; \mathcal{T}|_{w,x} \cup \mathcal{T}|_{p,x} \; , <_{PO})$$

$$\exists \; \blacktriangleleft = \mathit{SerialView}(\; \mathcal{T} \; , \qquad <_{PO}) \qquad [SC]$$

Hutto, Ahamad 1990

NP-hard for most models

Slow Consistency

Writes from one process to one variable are observed in same order by all processes.

- The program order is respected.
- For each process p and variable x: there exists a serial view on all writes to x and reads from x of p.

$$\forall x, p \; \exists \; \blacktriangleleft = SerialView(\; \mathcal{T}|_{w,x} \cup \mathcal{T}|_{p,x} \;, <_{PO})$$

$$\exists \; \blacktriangleleft = SerialView(\; \mathcal{T} \;, \; <_{PO}) \qquad [SC]$$

Hutto, Ahamad 1990

NP-hard for most models

$\mathrm{SC} \leq \mathrm{SLOW}$-Range-Reduction of SAT

Reduction Idea

- Test uses only one variable ξ.
- Test has only one process with reads.
- $\Rightarrow$ Test behaves the same from Slow to SC.

$$\forall x, p \; \exists \quad \blacktriangleleft = \mathit{SerialView}(\; \mathcal{T}|_{w,x} \cup \mathcal{T}|_{p,x} \; , <_{PO}) \qquad [\mathit{Slow}]$$

$$\exists \quad \blacktriangleleft = \mathit{SerialView}(\; \mathcal{T} \; , <_{PO}) \qquad [\mathit{SC}]$$

NP-hard for most models

SC ≤ SLOW-Range-Reduction of SAT

Reduction Idea

- Test uses only one variable ξ.
- Test has only one process with reads.
- $\Rightarrow$ Test behaves the same from Slow to SC.

$$\forall x, p\ \exists\ \blacktriangleleft = SerialView(\ \mathcal{T}|_{w,x} \cup \mathcal{T}|_{p,x}\ , <_{PO}) \qquad [Slow]$$
$$\exists\ \blacktriangleleft = SerialView(\ \mathcal{T}\ , <_{PO}) \qquad [SC]$$

SAT-Reduction

- We associate clauses and variables with values of ξ.

SC-Slow Reduction - Example

$$\underbrace{(a \vee b)}_{cl_1} \wedge \underbrace{\neg a}_{cl_2}$$

$a = false$
$b = true$

SC-Slow Reduction - Example

$$\underbrace{(a \vee b)}_{cl_1} \wedge \underbrace{\neg a}_{cl_2}$$

$a = false$
$b = true$

$$True_a := (w, \xi, cl1) \ . \ (w, \xi, a)$$
$$False_a := (w, \xi, cl2) \ . \ (w, \xi, a)$$

SC-Slow Reduction - Example

$$\underbrace{(a \vee b)}_{cl_1} \wedge \underbrace{\neg a}_{cl_2}$$

$a = false$
$b = true$

$$\begin{aligned}
True_a &:= (w, \xi, cl1)\ .\ (w, \xi, a) \\
False_a &:= (w, \xi, cl2)\ .\ (w, \xi, a) \\
True_b &:= (w, \xi, cl1)\ .\ (w, \xi, b) \\
False_b &:= (w, \xi, b)
\end{aligned}$$

SC-Slow Reduction - Example

$$\underbrace{(a \vee b)}_{cl_1} \wedge \underbrace{\neg a}_{cl_2}$$

$$a = \mathit{false}$$
$$b = \mathit{true}$$

$$\begin{aligned}
True_a &:= (w, \xi, cl1) \ . \ (w, \xi, a) \\
False_a &:= (w, \xi, cl2) \ . \ (w, \xi, a) \\
True_b &:= (w, \xi, cl1) \ . \ (w, \xi, b) \\
False_b &:= (w, \xi, b) \\
Eval &:= (r, \xi, a) \ . \ (r, \xi, b) \ . \ (r, \xi, cl1) \ . \ (r, \xi, cl2)
\end{aligned}$$

◀ :

ξ : 0

SC-Slow Reduction - Example

$$\underbrace{(a \vee b)}_{cl_1} \wedge \underbrace{\neg a}_{cl_2}$$

$$a = false$$
$$b = true$$

$$\begin{aligned} True_a &:= (w, \xi, cl1)\ .\ (w, \xi, a) \\ False_a &:= (w, \xi, cl2)\ .\ (w, \xi, a) \\ True_b &:= (w, \xi, cl1)\ .\ (w, \xi, b) \\ False_b &:= (w, \xi, b) \\ Eval &:= (r, \xi, a)\ .\ (r, \xi, b)\ .\ (r, \xi, cl1)\ .\ (r, \xi, cl2) \end{aligned}$$

$\blacktriangleleft :\quad (w, \xi, cl1)$

$\xi :\quad 1$

SC-Slow Reduction - Example

$$\underbrace{(a \vee b)}_{cl_1} \wedge \underbrace{\neg a}_{cl_2} \qquad\qquad a = \mathit{false}$$
$$b = \mathit{true}$$

$$
\begin{aligned}
True_a &:= (w, \xi, cl1)\ .\ (w, \xi, a) \\
False_a &:= (w, \xi, cl2)\ .\ (w, \xi, a) \\
True_b &:= (w, \xi, cl1)\ .\ (w, \xi, b) \\
False_b &:= (w, \xi, b) \\
Eval &:= (r, \xi, a)\ .\ (r, \xi, b)\ .\ (r, \xi, cl1)\ .\ (r, \xi, cl2)
\end{aligned}
$$

$\blacktriangleleft : \quad (w, \xi, cl1).(w, \xi, a)$

$\xi : \quad a$

SC-Slow Reduction - Example

$$\underbrace{(a \vee b)}_{cl_1} \wedge \underbrace{\neg a}_{cl_2} \qquad\qquad a = \mathit{false}$$
$$b = \mathit{true}$$

$$\begin{aligned}
True_a &:= (w, \xi, cl1)\ .\ (w, \xi, a) \\
False_a &:= (w, \xi, cl2)\ .\ (w, \xi, a) \\
True_b &:= (w, \xi, cl1)\ .\ (w, \xi, b) \\
False_b &:= (w, \xi, b) \\
Eval &:= (r, \xi, a)\ .\ (r, \xi, b)\ .\ (r, \xi, cl1)\ .\ (r, \xi, cl2)
\end{aligned}$$

$\blacktriangleleft :\quad (w, \xi, cl1).(w, \xi, a).(r, \xi, a)$

$\xi :\quad a$

SC-Slow Reduction - Example

$$\underbrace{(a \vee b)}_{cl_1} \wedge \underbrace{\neg a}_{cl_2} \qquad a = false \quad b = true$$

$$\begin{aligned}
True_a &:= (w, \xi, cl1)\ .\ (w, \xi, a) \\
False_a &:= (w, \xi, cl2)\ .\ (w, \xi, a) \\
True_b &:= (w, \xi, cl1)\ .\ (w, \xi, b) \\
False_b &:= (w, \xi, b) \\
Eval &:= (r, \xi, a)\ .\ (r, \xi, b)\ .\ (r, \xi, cl1)\ .\ (r, \xi, cl2)
\end{aligned}$$

$\blacktriangleleft$: $(w, \xi, cl1).(w, \xi, a).(r, \xi, a).(w, \xi, b)$

ξ : b

SC-Slow Reduction - Example

$$\underbrace{(a \vee b)}_{cl_1} \wedge \underbrace{\neg a}_{cl_2} \qquad\qquad a = \mathit{false}$$
$$b = \mathit{true}$$

$$\begin{aligned}
\mathit{True}_a &:= (w, \xi, cl1)\ .\ (w, \xi, a) \\
\mathit{False}_a &:= (w, \xi, cl2)\ .\ (w, \xi, a) \\
\mathit{True}_b &:= (w, \xi, cl1)\ .\ (w, \xi, b) \\
\mathit{False}_b &:= (w, \xi, b) \\
\mathit{Eval} &:= (r, \xi, a)\ .\ (r, \xi, b)\ .\ (r, \xi, cl1)\ .\ (r, \xi, cl2)
\end{aligned}$$

$\blacktriangleleft :\quad (w, \xi, cl1).(w, \xi, a).(r, \xi, a).(w, \xi, b).(r, \xi, b)$

$\xi :\quad b$

SC-Slow Reduction - Example

$$\underbrace{(a \vee b)}_{cl_1} \wedge \underbrace{\neg a}_{cl_2}$$

$a = \mathit{false}$
$b = \mathit{true}$

$$\begin{aligned}
True_a &:= (w, \xi, cl1) \;.\; (w, \xi, a) \\
False_a &:= (w, \xi, cl2) \;.\; (w, \xi, a) \\
True_b &:= (w, \xi, cl1) \;.\; (w, \xi, b) \\
False_b &:= (w, \xi, b) \\
Eval &:= (r, \xi, a) \;.\; (r, \xi, b) \;.\; (r, \xi, cl1) \;.\; (r, \xi, cl2)
\end{aligned}$$

◀ : $(w, \xi, cl1).(w, \xi, a).(r, \xi, a).(w, \xi, b).(r, \xi, b)$
$.(w, \xi, cl1)$

ξ : 1

SC-Slow Reduction - Example

$$\underbrace{(a \vee b)}_{cl_1} \wedge \underbrace{\neg a}_{cl_2} \qquad \begin{array}{l} a = \mathit{false} \\ b = \mathit{true} \end{array}$$

$$\begin{aligned}
True_a &:= (w, \xi, cl1) \ . \ (w, \xi, a) \\
False_a &:= (w, \xi, cl2) \ . \ (w, \xi, a) \\
True_b &:= (w, \xi, cl1) \ . \ (w, \xi, b) \\
False_b &:= (w, \xi, b) \\
Eval &:= (r, \xi, a) \ . \ (r, \xi, b) \ . \ (r, \xi, cl1) \ . \ (r, \xi, cl2)
\end{aligned}$$

$$\begin{array}{rl}
\blacktriangleleft : & (w, \xi, cl1).(w, \xi, a).(r, \xi, a).(w, \xi, b).(r, \xi, b) \\
 & .(w, \xi, cl1).(r, \xi, cl1) \\
\xi : & 1
\end{array}$$

SC-Slow Reduction - Example

$$\underbrace{(a \vee b)}_{cl_1} \wedge \underbrace{\neg a}_{cl_2}$$

$$a = false$$
$$b = true$$

$$\begin{aligned}
True_a &:= (w, \xi, cl1) \ . \ (w, \xi, a) \\
False_a &:= (w, \xi, cl2) \ . \ (w, \xi, a) \\
True_b &:= (w, \xi, cl1) \ . \ (w, \xi, b) \\
False_b &:= (w, \xi, b) \\
Eval &:= (r, \xi, a) \ . \ (r, \xi, b) \ . \ (r, \xi, cl1) \ . \ (r, \xi, cl2)
\end{aligned}$$

$\blacktriangleleft$: $(w, \xi, cl1).(w, \xi, a).(r, \xi, a).(w, \xi, b).(r, \xi, b)$
$.(w, \xi, cl1).(r, \xi, cl1).(w, \xi, cl2)$

ξ : 2

SC-Slow Reduction - Example

$$\underbrace{(a \vee b)}_{cl_1} \wedge \underbrace{\neg a}_{cl_2} \qquad\qquad a = \mathit{false} \quad b = \mathit{true}$$

$$\begin{aligned}
True_a &:= (w, \xi, cl1)\ .\ (w, \xi, a) \\
False_a &:= (w, \xi, cl2)\ .\ (w, \xi, a) \\
True_b &:= (w, \xi, cl1)\ .\ (w, \xi, b) \\
False_b &:= (w, \xi, b) \\
Eval &:= (r, \xi, a)\ .\ (r, \xi, b)\ .\ (r, \xi, cl1)\ .\ (r, \xi, cl2)
\end{aligned}$$

$$\begin{aligned}
\blacktriangleleft:\quad & (w, \xi, cl1).(w, \xi, a).(r, \xi, a).(w, \xi, b).(r, \xi, b) \\
& .(w, \xi, cl1).(r, \xi, cl1).(w, \xi, cl2).(r, \xi, cl2) \\
\xi:\quad & 2
\end{aligned}$$

SC-Slow Reduction - Example

$$\underbrace{(a \vee b)}_{cl_1} \wedge \underbrace{\neg a}_{cl_2} \qquad\qquad a = \mathit{false}$$
$$b = \mathit{true}$$

$$\begin{aligned}
True_a &:= (w, \xi, cl1) \;.\; (w, \xi, a) \\
False_a &:= (w, \xi, cl2) \;.\; (w, \xi, a) \\
True_b &:= (w, \xi, cl1) \;.\; (w, \xi, b) \\
False_b &:= (w, \xi, b) \\
Eval &:= (r, \xi, a) \;.\; (r, \xi, b) \;.\; (r, \xi, cl1) \;.\; (r, \xi, cl2)
\end{aligned}$$

$$\begin{aligned}
\blacktriangleleft :\quad & (w, \xi, cl1).(w, \xi, a).(r, \xi, a).(w, \xi, b).(r, \xi, b) \\
& .(w, \xi, cl1).(r, \xi, cl1).(w, \xi, cl2).(r, \xi, cl2).(w, \xi, a) \\
\xi :\quad & a
\end{aligned}$$

SC-Slow Reduction - Example

$$\underbrace{(a \vee b)}_{cl_1} \wedge \underbrace{\neg a}_{cl_2}$$

$a = false$
$b = true$

$$True_a := (w, \xi, cl1)\ .\ (w, \xi, a)$$
$$False_a := (w, \xi, cl2)\ .\ (w, \xi, a)$$
$$True_b := (w, \xi, cl1)\ .\ (w, \xi, b)$$
$$False_b := (w, \xi, b)$$
$$Eval := (r, \xi, a)\ .\ (r, \xi, b)\ .\ (r, \xi, cl1)\ .\ (r, \xi, cl2)$$

$\blacktriangleleft$: $(w, \xi, cl1).(w, \xi, a).(r, \xi, a).(w, \xi, b).(r, \xi, b)$
$.(w, \xi, cl1).(r, \xi, cl1).(w, \xi, cl2).(r, \xi, cl2).(w, \xi, a).(w, \xi, b)$

ξ : b

SC-Slow Reduction - Example

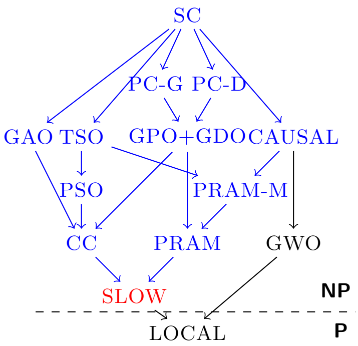
$$\underbrace{(a \vee b)}_{cl_1} \wedge \underbrace{\neg a}_{cl_2} \qquad\qquad a = \mathit{false}$$
$$b = \mathit{true}$$

$$\begin{aligned}
True_a &:= (w, \xi, cl1)\ .\ (w, \xi, a) \\
False_a &:= (w, \xi, cl2)\ .\ (w, \xi, a) \\
True_b &:= (w, \xi, cl1)\ .\ (w, \xi, b) \\
False_b &:= (w, \xi, b) \\
Eval &:= (r, \xi, a)\ .\ (r, \xi, b)\ .\ (r, \xi, cl1)\ .\ (r, \xi, cl2)
\end{aligned}$$

$$\begin{aligned}
\blacktriangleleft : &\quad (w, \xi, cl1).(w, \xi, a).(r, \xi, a).(w, \xi, b).(r, \xi, b) \\
&\quad .(w, \xi, cl1).(r, \xi, cl1).(w, \xi, cl2).(r, \xi, cl2).(w, \xi, a).(w, \xi, b) \\
\xi : &\quad b
\end{aligned}$$

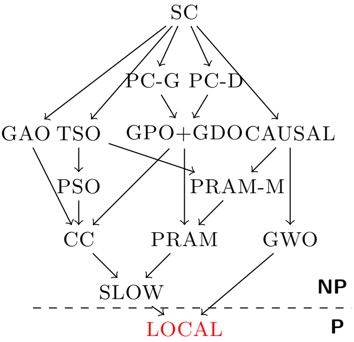
Results

Memory Model	Complexity Class of TEST(M)			
	General	Process	Length	Variables
SC	**NPC**(by 1)			**NPC**(by 1)
TSO	**NPC**(by 1)			**NPC**(by 1)
PSO	**NPC**(by 1)			**NPC**(by 1)
PC-G	**NPC**(by 1)			**NPC**(by 1)
PC-D	**NPC**(by 1)			**NPC**(by 1)
GAO	**NPC**(by 1)			**NPC**(by 1)
GPO+GDO	**NPC**(by 1)			**NPC**(by 1)
Causal	**NPC**(by 1)			**NPC**(by 1)
PRAM-M	**NPC**(by 1)			**NPC**(by 1)
GWO				
CC	**NPC**(by 1)			**NPC**(by 1)
PRAM	**NPC**(by 1)			**NPC**(by 1)
SLOW	**NPC**(by 1)			**NPC**$_1$
LOCAL				



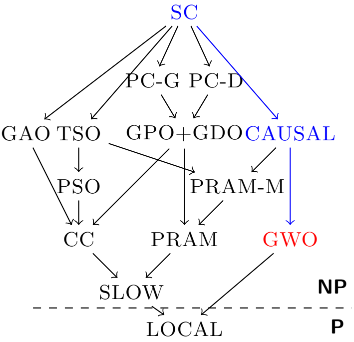
Results

Memory Model	Complexity Class of TEST(M)			
	General	Process	Length	Variables
SC	**NPC**(by 1)			**NPC**(by 1)
TSO	**NPC**(by 1)			**NPC**(by 1)
PSO	**NPC**(by 1)			**NPC**(by 1)
PC-G	**NPC**(by 1)			**NPC**(by 1)
PC-D	**NPC**(by 1)			**NPC**(by 1)
GAO	**NPC**(by 1)			**NPC**(by 1)
GPO+GDO	**NPC**(by 1)			**NPC**(by 1)
Causal	**NPC**(by 1)			**NPC**(by 1)
PRAM-M	**NPC**(by 1)			**NPC**(by 1)
GWO				
CC	**NPC**(by 1)			**NPC**(by 1)
PRAM	**NPC**(by 1)			**NPC**(by 1)
SLOW	**NPC**(by 1)			$\mathbf{NPC}_1$
LOCAL	$\mathbf{P}_2$	**P**(by 2)	**P**(by 2)	**P**(by 2)



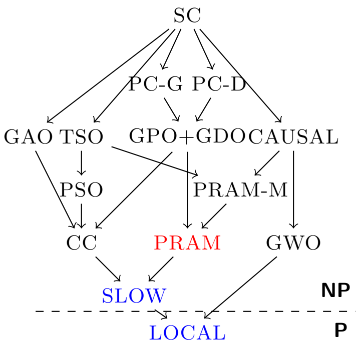
Results

Memory Model	Complexity Class of TEST(M)			
	General	Process	Length	Variables
SC	**NPC**(by 1)		**NPC**(by 3)	**NPC**(by 1)
TSO	**NPC**(by 1)			**NPC**(by 1)
PSO	**NPC**(by 1)			**NPC**(by 1)
PC-G	**NPC**(by 1)			**NPC**(by 1)
PC-D	**NPC**(by 1)			**NPC**(by 1)
GAO	**NPC**(by 1)			**NPC**(by 1)
GPO+GDO	**NPC**(by 1)			**NPC**(by 1)
Causal	**NPC**(by 1)		**NPC**(by 3)	**NPC**(by 1)
PRAM-M	**NPC**(by 1)			**NPC**(by 1)
GWO	**NPC**(by 3)		$\mathbf{NPC}_3$	
CC	**NPC**(by 1)			**NPC**(by 1)
PRAM	**NPC**(by 1)			**NPC**(by 1)
SLOW	**NPC**(by 1)			$\mathbf{NPC}_1$
LOCAL	$\mathbf{P}_2$	**P**(by 2)	**P**(by 2)	**P**(by 2)



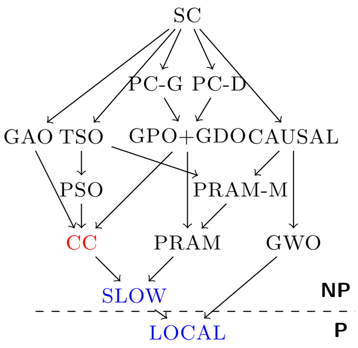
Results

Memory Model	Complexity Class of TEST(M)			
	General	Process	Length	Variables
SC	**NPC**(by 1)		**NPC**(by 3)	**NPC**(by 1)
TSO	**NPC**(by 1)			**NPC**(by 1)
PSO	**NPC**(by 1)			**NPC**(by 1)
PC-G	**NPC**(by 1)			**NPC**(by 1)
PC-D	**NPC**(by 1)			**NPC**(by 1)
GAO	**NPC**(by 1)			**NPC**(by 1)
GPO+GDO	**NPC**(by 1)			**NPC**(by 1)
Causal	**NPC**(by 1)		**NPC**(by 3)	**NPC**(by 1)
PRAM-M	**NPC**(by 1)			**NPC**(by 1)
GWO	**NPC**(by 3)		$\mathbf{NPC}_3$	
CC	**NPC**(by 1)			**NPC**(by 1)
PRAM	**NPC**(by 1)		$\mathbf{P}_4$	**NPC**(by 1)
SLOW	**NPC**(by 1)		**P**(by 4)	$\mathbf{NPC}_1$
LOCAL	$\mathbf{P}_2$	**P**(by 2)	**P**(by 2)	**P**(by 2)



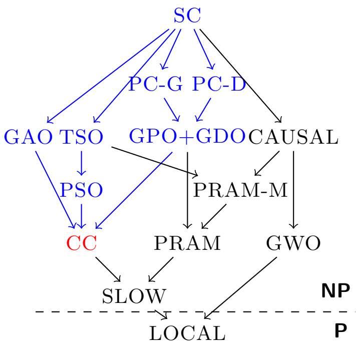
Results

Memory Model	Complexity Class of TEST(M)			
	General	Process	Length	Variables
SC	**NPC**(by 1)		**NPC**(by 3)	**NPC**(by 1)
TSO	**NPC**(by 1)			**NPC**(by 1)
PSO	**NPC**(by 1)			**NPC**(by 1)
PC-G	**NPC**(by 1)			**NPC**(by 1)
PC-D	**NPC**(by 1)			**NPC**(by 1)
GAO	**NPC**(by 1)			**NPC**(by 1)
GPO+GDO	**NPC**(by 1)			**NPC**(by 1)
Causal	**NPC**(by 1)		**NPC**(by 3)	**NPC**(by 1)
PRAM-M	**NPC**(by 1)			**NPC**(by 1)
GWO	**NPC**(by 3)		$\mathbf{NPC}_3$	
CC	**NPC**(by 1)	$\mathbf{P}_5$		**NPC**(by 1)
PRAM	**NPC**(by 1)		$\mathbf{P}_4$	**NPC**(by 1)
SLOW	**NPC**(by 1)	**P**(by 5)	**P**(by 4)	$\mathbf{NPC}_1$
LOCAL	$\mathbf{P}_2$	**P**(by 2)	**P**(by 2)	**P**(by 2)



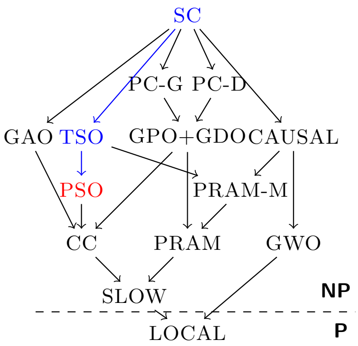
Results

Memory Model	Complexity Class of TEST(M)			
	General	Process	Length	Variables
SC	**NPC**(by 1)		**NPC**(by 3)	**NPC**(by 1)
TSO	**NPC**(by 1)		**NPC**(by 6)	**NPC**(by 1)
PSO	**NPC**(by 1)		**NPC**(by 6)	**NPC**(by 1)
PC-G	**NPC**(by 1)		**NPC**(by 6)	**NPC**(by 1)
PC-D	**NPC**(by 1)		**NPC**(by 6)	**NPC**(by 1)
GAO	**NPC**(by 1)		**NPC**(by 6)	**NPC**(by 1)
GPO+GDO	**NPC**(by 1)		**NPC**(by 6)	**NPC**(by 1)
Causal	**NPC**(by 1)		**NPC**(by 3)	**NPC**(by 1)
PRAM-M	**NPC**(by 1)			**NPC**(by 1)
GWO	**NPC**(by 3)		$\mathbf{NPC}_3$	
CC	**NPC**(by 1)	$\mathbf{P}_5$	$\mathbf{NPC}_6$	**NPC**(by 1)
PRAM	**NPC**(by 1)		$\mathbf{P}_4$	**NPC**(by 1)
SLOW	**NPC**(by 1)	**P**(by 5)	**P**(by 4)	$\mathbf{NPC}_1$
LOCAL	$\mathbf{P}_2$	**P**(by 2)	**P**(by 2)	**P**(by 2)



Results

Memory Model	Complexity Class of TEST(M)			
	General	Process	Length	Variables
SC	**NPC**(by 1)	**NPC**(by 7)	**NPC**(by 3)	**NPC**(by 1)
TSO	**NPC**(by 1)	**NPC**(by 7)	**NPC**(by 6)	**NPC**(by 1)
PSO	**NPC**(by 1)	$\mathbf{NPC}_7$	**NPC**(by 6)	**NPC**(by 1)
PC-G	**NPC**(by 1)		**NPC**(by 6)	**NPC**(by 1)
PC-D	**NPC**(by 1)		**NPC**(by 6)	**NPC**(by 1)
GAO	**NPC**(by 1)		**NPC**(by 6)	**NPC**(by 1)
GPO+GDO	**NPC**(by 1)		**NPC**(by 6)	**NPC**(by 1)
Causal	**NPC**(by 1)		**NPC**(by 3)	**NPC**(by 1)
PRAM-M	**NPC**(by 1)			**NPC**(by 1)
GWO	**NPC**(by 3)		$\mathbf{NPC}_3$	
CC	**NPC**(by 1)	$\mathbf{P}_5$	$\mathbf{NPC}_6$	**NPC**(by 1)
PRAM	**NPC**(by 1)		$\mathbf{P}_4$	**NPC**(by 1)
SLOW	**NPC**(by 1)	**P**(by 5)	**P**(by 4)	$\mathbf{NPC}_1$
LOCAL	$\mathbf{P}_2$	**P**(by 2)	**P**(by 2)	**P**(by 2)



Results

Memory Model	Complexity Class of TEST(M)			
	General	Process	Length	Variables
SC	**NPC**(by 1)	**NPC**(by 7)	**NPC**(by 3)	**NPC**(by 1)
TSO	**NPC**(by 1)	**NPC**(by 7)	**NPC**(by 6)	**NPC**(by 1)
PSO	**NPC**(by 1)	$\mathbf{NPC}_7$	**NPC**(by 6)	**NPC**(by 1)
PC-G	**NPC**(by 1)		**NPC**(by 6)	**NPC**(by 1)
PC-D	**NPC**(by 1)		**NPC**(by 6)	**NPC**(by 1)
GAO	**NPC**(by 1)		**NPC**(by 6)	**NPC**(by 1)
GPO+GDO	**NPC**(by 1)		**NPC**(by 6)	**NPC**(by 1)
Causal	**NPC**(by 1)		**NPC**(by 3)	**NPC**(by 1)
PRAM-M	**NPC**(by 1)			**NPC**(by 1)
GWO	**NPC**(by 3)		$\mathbf{NPC}_3$	
CC	**NPC**(by 1)	$\mathbf{P}_5$	$\mathbf{NPC}_6$	**NPC**(by 1)
PRAM	**NPC**(by 1)		$\mathbf{P}_4$	**NPC**(by 1)
SLOW	**NPC**(by 1)	**P**(by 5)	**P**(by 4)	$\mathbf{NPC}_1$
LOCAL	$\mathbf{P}_2$	**P**(by 2)	**P**(by 2)	**P**(by 2)