

Weak Memory Models: an Operational Theory

G erard Boudol and Gustavo Petri

INRIA Sophia Antipolis

9th June 2008

Memory models, what are they good for?

- Hardware optimizations
- Contract between hardware and software
- Defines which values can be read from the memory
- **Semantics of concurrency**

Memory models, what do they specify?

- Define memory actions (events)
- Ordering constraints
- Visibility constraints
- Atomicity constraints
- Determines allowed optimizations

Memory models, how are they specified?

- (Lamport 79)

*... result of any execution is the same as if the **operations** of all the processors where executed ...*

- (Java Memory Model 2005)

*An **action** a is described by a tuple $\langle t, k, v, u \rangle$, comprising: t - the thread performing the action ...*

- (Intel 64 2007)

Stores are not reordered with older loads

Processor 0	Processor 1
<i>mov r1, [- x] // M1</i>	<i>mov r2, [- y] // M3</i>
<i>mov [- y], 1 // M2</i>	<i>mov [- x], 1 // M4</i>
<i>Initially $x == y == 0$</i>	
<i>$r1 == 1$ and $r1 == 1$ is not allowed</i>	

An operational theory

Standard memory models build around

- single processor ordering
- **happens before** order (Lamport78)
- rules to restrict reordering of instructions

Our proposal: **Standard** Operational Semantics Techniques

- a small step interleaving semantics
- a small step **weak** semantics (write buffers)
- **true concurrency** techniques to define: events, conflict, concurrency, dependency
- **bisimulation** to prove DRF guarantee

A simple calculus

$$e ::= v \mid (e_0 e_1) \quad (\text{expressions})$$

$$\quad \mid (\text{ref } e) \mid (!e) \mid (e_0 := e_1)$$

$$\quad \mid (\text{thread } e) \mid (\text{with } l \text{ do } e)$$

$$v ::= x \mid (\lambda x e) \mid () \quad (\text{values})$$

$$r ::= (\lambda x e_0 e_1) \mid (!a) \mid (a := v) \quad (\text{redexes})$$

$$\mathbf{E} ::= [] \mid \mathbf{E}[\mathbf{F}] \quad (\text{evaluation contexts})$$

$$\mathbf{F} ::= ([] e) \mid (v []) \quad (\text{frames})$$

$$\quad \mid (\text{ref} []) \mid (![]) \mid ([] := e)$$

$$\quad \mid (v := []) \mid (\text{holding } l \text{ do } [])$$

Strong Semantics: Interleaving

$$(S, L, \mathbf{T[E}[(\lambda xev)])]) \longrightarrow (S, L, \mathbf{T[E}[\{x \mapsto v\}e_0]])$$

$$(S, L, \mathbf{T[E}[(\text{ref } v)])]) \longrightarrow (S\{p \mapsto v\}, L, \mathbf{T[E}[x]]) \quad p \notin \text{dom}(S)$$

$$(S, L, \mathbf{T[E}[(!p)])]) \longrightarrow (S, L, \mathbf{T[E}[v]]) \quad S(p) = v$$

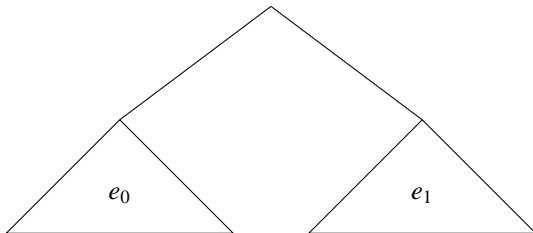
$$(S, L, \mathbf{T[E}[(p := v)])]) \longrightarrow (S\{p \mapsto v\}, L, \mathbf{T[E}[(\text{ })]])$$

$$(S, L, \mathbf{T[E}[(\text{thread } e)])]) \longrightarrow (S, L, \mathbf{T[E}[(\mathbf{E}[(\text{ })] \parallel e)])])$$

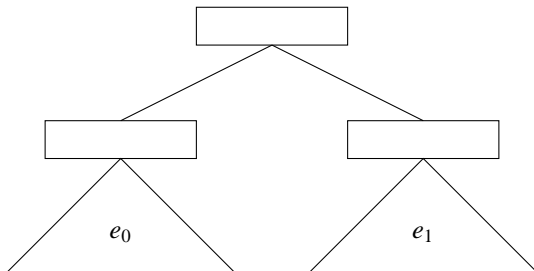
$$(S, L, \mathbf{T[E}[(\text{with } l \text{ do } e)])]) \longrightarrow (S, L \cup \{l\}, \mathbf{T[E}[(\text{holding } l \text{ do } e)])]) \quad l \notin L$$

$$(S, L, \mathbf{T[E}[(\text{holding } l \text{ do } v)])]) \longrightarrow (S, L - \{l\}, \mathbf{T[E}[v]])$$

Adding Write Buffers



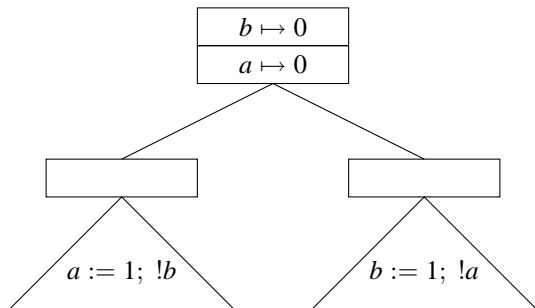
Adding Write Buffers



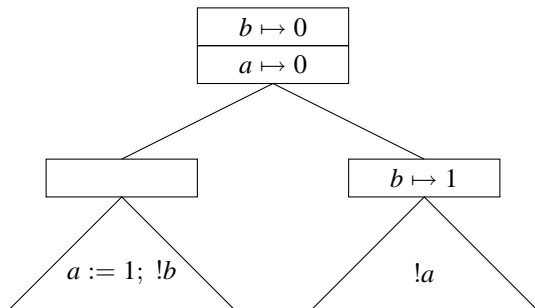
Weak Semantics: Write buffers

$$\begin{array}{lcl}
 (S, L, \mathbf{B}[\mathbf{E}[(!p)]) & \rightsquigarrow & (S, L, \mathbf{B}[\mathbf{E}[v]]) \\
 (S, L, \mathbf{B}[\mathbf{E}[(p := v)]) & \rightsquigarrow & \mathbf{B}(p) = \epsilon, S(p) = v \\
 & & (S, L, \mathbf{B}[\mathbf{E}[(\{p \mapsto v\}, \mathbf{E}[\cdot])]]) \\
 (S, L, \mathbf{B}[\mathbf{E}[(\text{holding } l \text{ do } v)]]) & \rightsquigarrow & (S, L - \{l\}, \mathbf{B}[\mathbf{E}[v]]) \\
 (S, L, \mathbf{B}[(b, (b', B))]) & \rightsquigarrow & \mathbf{B}^\dagger \\
 & & (S, L, \mathbf{B}[(\text{put}(b, p, v), (\text{pop}(b', p), B))]) \\
 & & p \in \text{dom}(b'), b'(p) = v.q \\
 (S, L, (b, B)) & \rightsquigarrow & (S\{p \mapsto v\}, L, (\text{pop}(b, x), B)) \\
 & & p \in \text{dom}(b), b(p) = v.q
 \end{array}$$

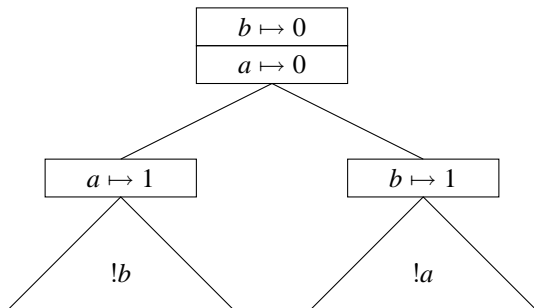
Racy example: Weak behavior

$$a := 1; !b \parallel b := 1; !a$$


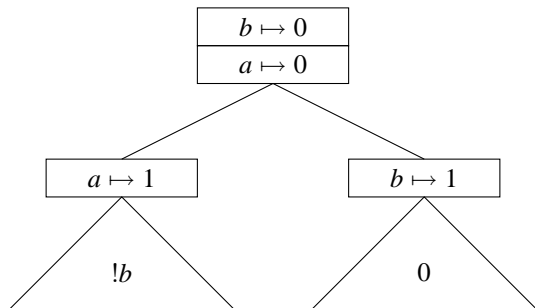
Racy example: Weak behavior

$$a := 1; !b \parallel b := 1; !a$$


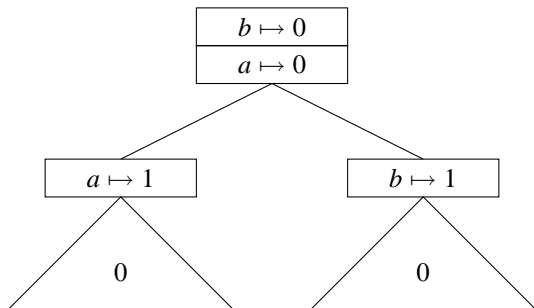
Racy example: Weak behavior

$$a := 1; !b \parallel b := 1; !a$$


Racy example: Weak behavior

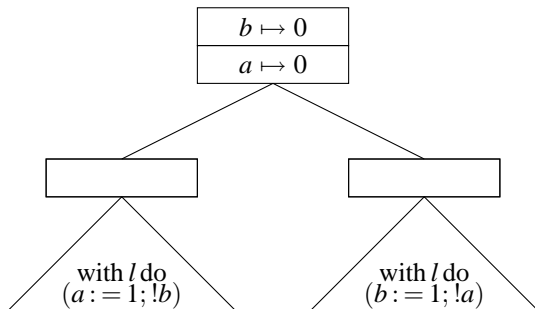
$$a := 1; !b \parallel b := 1; !a$$


Racy example: Weak behavior

$$a := 1; !b \parallel b := 1; !a$$


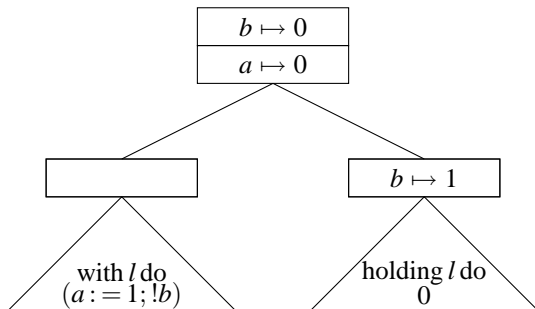
Correctly Synchronized: Strong behavior

with l do($a := 1; !b$) || with l do($b := 1; !a$)



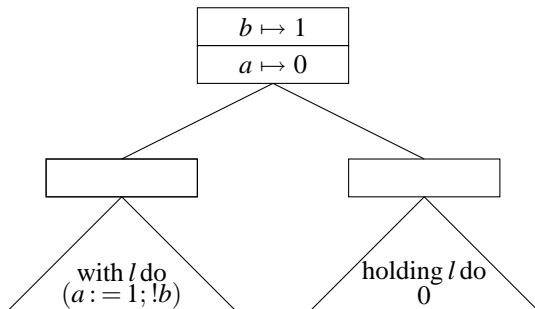
Correctly Synchronized: Strong behavior

with l do($a := 1; !b$) || with l do($b := 1; !a$)



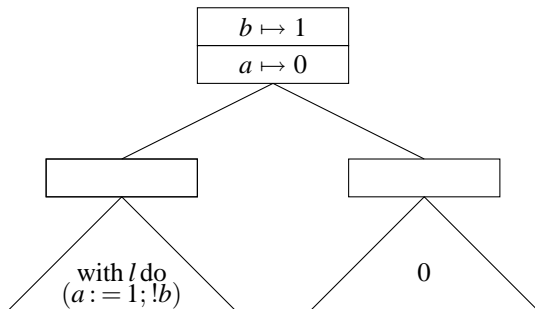
Correctly Synchronized: Strong behavior

with l do($a := 1; !b$) || with l do($b := 1; !a$)



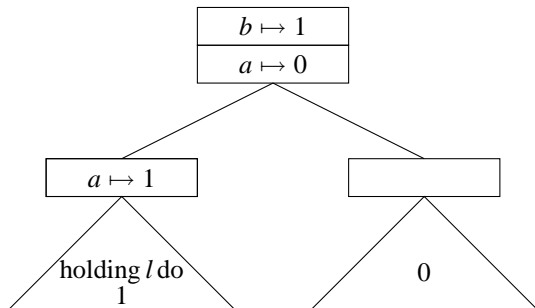
Correctly Synchronized: Strong behavior

with l do($a := 1; !b$) || with l do($b := 1; !a$)



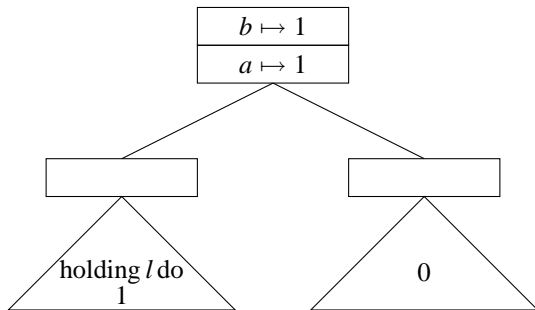
Correctly Synchronized: Strong behavior

with $l \text{ do}(a := 1; !b) \parallel \text{with } l \text{ do}(b := 1; !a)$



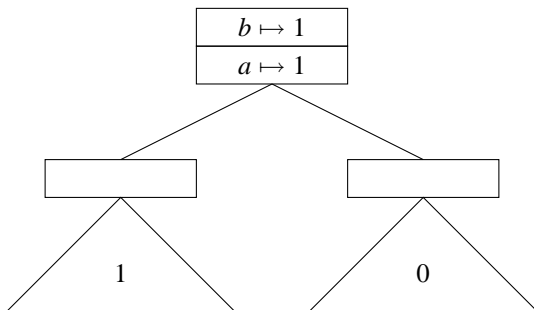
Correctly Synchronized: Strong behavior

with $l \text{ do}(a := 1; !b) \parallel \text{with } l \text{ do}(b := 1; !a)$



Correctly Synchronized: Strong behavior

with l do($a := 1; !b$) \parallel with l do($b := 1; !a$)



Dekker's mutual exclusion

$$\begin{array}{l} \textit{flag}_1 := \textit{ff}; \\ \text{if } (!\textit{flag}_2) \text{ then} \\ \quad \textit{Critical Section} \end{array} \parallel \parallel \begin{array}{l} \textit{flag}_2 := \textit{ff}; \\ \text{if } (!\textit{flag}_1) \text{ then} \\ \quad \textit{Critical Section} \end{array}$$

Dekker's mutual exclusion

$$\begin{array}{l}
 \textit{flag}_1 := \textit{ff}; \\
 \text{if } (!\textit{flag}_2) \text{ then} \\
 \quad \textit{Critical Section}
 \end{array}
 \parallel
 \begin{array}{l}
 \textit{flag}_2 := \textit{ff}; \\
 \text{if } (!\textit{flag}_1) \text{ then} \\
 \quad \textit{Critical Section}
 \end{array}$$

Not Safe

Publication

$$\begin{array}{l} data := 8; \\ flag_1 := ff \end{array} \parallel \begin{array}{l} \text{if } (!flag_1) \text{ then} \\ r_1 := (!data) \end{array}$$

Publication

$$\begin{array}{l} data := 8; \\ flag_1 := ff \end{array} \parallel \begin{array}{l} \text{if } (!flag_1) \text{ then} \\ r_1 := (!data) \end{array}$$

Not safe: $r_1 \neq 8$

Some definitions

- **Correctly Synchronized** A program is correctly synchronized if all its **strong executions** are free of data races
- **DRF Guarantee** If a program is correctly synchronized, all its behaviors in the weak semantics are sequentially consistent
- **Our approach to DRF** The strong and weak semantics are **bisimilar** for correctly synchronized programs

Correctly Synchronized Programs

CS key property

Concurrent conflicting events are **dependent** in every execution.

So we need to define:

- Event
- Conflict of events ($\#$)
- Concurrency of events (\sphericalcap)
- Dependency of events (\prec)

True concurrency: Event

- We want to derive events from the semantics
- Annotate the semantics with actions
 - Action type
 - Who performed the action

Strong Semantics: annotated

$$(S, L, \mathbf{T}[\mathbf{E}[(\lambda xev)]]) \xrightarrow[\textcircled{\mathbf{T}}]{\beta} (S, L, \mathbf{T}[\mathbf{E}[\{x \mapsto v\}e_0]])$$

$$(S, L, \mathbf{T}[\mathbf{E}[(\text{ref } v)]]) \xrightarrow[\textcircled{\mathbf{T}}]{\nu_p} (S\{p \mapsto v\}, L, \mathbf{T}[\mathbf{E}[x]]) \quad p \notin \text{dom}(S)$$

$$(S, L, \mathbf{T}[\mathbf{E}[(!x)]]) \xrightarrow[\textcircled{\mathbf{T}}]{\text{rd}_p} (S, L, \mathbf{T}[\mathbf{E}[v]]) \quad S(p) = v$$

$$(S, L, \mathbf{T}[\mathbf{E}[(p := v)]]) \xrightarrow[\textcircled{\mathbf{T}}]{\text{wr}_p} (S\{p \mapsto v\}, L, \mathbf{T}[\mathbf{E}[(\)]])$$

$$(S, L, \mathbf{T}[\mathbf{E}[(\text{thread } e)]]) \xrightarrow[\textcircled{\mathbf{T}}]{\text{spw}} (S, L, \mathbf{T}[(\mathbf{E}[(\)] \parallel e)])$$

$$(S, L, \mathbf{T}[\mathbf{E}[(\text{with } l \text{ do } e)]]) \xrightarrow[\textcircled{\mathbf{T}}]{\hat{l}} (S, L \cup \{l\}, \mathbf{T}[\mathbf{E}[(\text{holding } l \text{ do } e)]]) \quad l \notin L$$

$$(S, L, \mathbf{T}[\mathbf{E}[(\text{holding } l \text{ do } v)]]) \xrightarrow[\textcircled{\mathbf{T}}]{\hat{l}} (S, L - \{l\}, \mathbf{T}[\mathbf{E}[v]])$$

Weak Semantics: annotated

$$\begin{array}{lcl}
(S, L, \mathbf{B}[\mathbf{E}[(!p)]) & \begin{array}{c} \text{rd}_p \\ \rightsquigarrow \\ @\mathbf{B} \end{array} & (S, L, \mathbf{B}[\mathbf{E}[v]]) \\
& & [\mathbf{B}](p) = \epsilon, S(p) = v \\
(S, L, \mathbf{B}[\mathbf{E}[(p := v)]) & \begin{array}{c} \text{wr}_p \\ \rightsquigarrow \\ @\mathbf{B} \end{array} & (S, L, \mathbf{B}[\mathbf{E}[(\{p \mapsto v\}, \mathbf{E}[\cdot])]]) \\
(S, L, \mathbf{B}[\mathbf{E}[(\text{holding } l \text{ do } v)]) & \begin{array}{c} \curvearrowright \\ \rightsquigarrow \\ @\mathbf{B} \end{array} & (S, L - \{l\}, \mathbf{B}[\mathbf{E}[v]]) \\
(S, L, \mathbf{B}[(b, (b', B))]) & \begin{array}{c} \mathbf{B}\dagger \\ \rightsquigarrow \\ @\mathbf{B} \end{array} & (S, L, \mathbf{B}[(\text{put}(b, p, v), (\text{pop}(b', p), B))]) \\
& & p \in \text{dom}(b'), b'(p) = v.q \\
(S, L, (b, B)) & \rightsquigarrow & (S\{p \mapsto v\}, L, (\text{pop}(b, x), B)) \\
& & p \in \text{dom}(b), b(p) = v.q
\end{array}$$

True concurrency: Event

- We derive events from the semantics
- Annotate the semantics with actions
 - Action type
 - Which thread performed it
- Events: $(a, o)_i$
 - a is the action
 - o is the occurrence in the pool (i.e. Thread ID)
 - i is the number of repetitions of that action in the execution so far

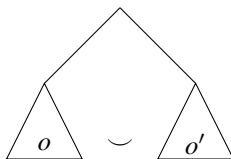
True concurrency: Conflict

Standard definition:

- Actions on the same memory location (at least one write)
 - $w_{r_p} \# rd_p$
 - $w_{r_p} \# w_{r_p}$
- Locking actions on the same lock
 - $\widehat{l} \# \widehat{l}$
 - $\widehat{l} \# \widehat{l}$
 - $\widehat{l} \# \widehat{l}$

True concurrency: Concurrency

Two occurrences are concurrent if neither is prefix of the other: $o \smile o'$



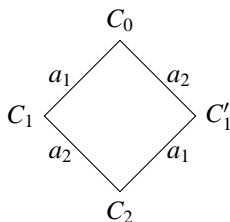
True concurrency: Asynchrony

If we have two events $(a, o)_i, (a, o)_j$ such that

- $C \xrightarrow[o_1]{a_1} C_1 \xrightarrow[o_2]{a_2} C_2$
- a_1 and a_2 are not conflicting
- o_1 and o_2 are concurrent

then there is a unique configuration C'_1 such that

- $C \xrightarrow[o_2]{a_2} C'_1 \xrightarrow[o_1]{a_1} C_2$



True concurrency: Asynchrony

If we have two events $(a, o)_i, (a, o)_j$ such that

- $C \xrightarrow{o_1} C_1 \xrightarrow{o_2} C_2$
- a_1 and a_2 are not conflicting
- o_1 and o_2 are concurrent

then there is a unique configuration C'_1 such that

- $C \xrightarrow{o_2} C'_1 \xrightarrow{o_1} C_2$

Equivalence by Permutation: $C \xrightarrow{o_1} C_1 \xrightarrow{o_2} C_2 \simeq C \xrightarrow{o_2} C'_1 \xrightarrow{o_1} C_2$

True concurrency: Ordering

$$(a, o)_i \leq_{\gamma} (a', o')_j$$

In any execution γ' such that $\gamma' \simeq \gamma$, we have that $(a, o)_i$ precedes $(a', o')_j$.

The DRF guarantee

- We define all these concepts for both semantics
- We define a relation between configurations
- We prove that our relation is a bisimulation for correctly synchronized programs

Conclusions

- A strong and a **weak Small Step Operational** semantics
- Instantiation of true concurrency definition to memory models
- **Bisimulation** proof of DRF for the weak semantics

Interesting observation

- The proof carries over if we allow **reading from buffers** (Cache)

Future work

- Formalize the proofs in Coq
- Experiment with different flavors of the weak semantics