Transactional Memories: a theoretical introduction

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Selim Arsever & Pascal Perez

Shared Memory Problems

$$M(w,r,v) := w(f).M\langle w,r,f\rangle + \underline{r}\langle v\rangle.M\langle w,r,v\rangle$$

$$A(w,r) := \underline{w}\langle v\rangle.WORK.r(v').[v'=v].A$$

$$\underline{A}\langle w,r\rangle \mid \underline{A}\langle w,r\rangle \mid M\langle w,r,v\rangle =$$

$$\underline{w}\langle v\rangle.WORK.r(v').[v'=v].A \mid$$

$$\underline{w}\langle v\rangle.WORK.r(v').[v'=v].A \mid$$

$$w(f).M\langle w,r,f\rangle + \underline{r}\langle v\rangle.M\langle w,r,v\rangle$$

Shared Memory Problems

$$M(w,r,v) := w(f).M\langle w,r,f\rangle + \underline{r}\langle v\rangle.M\langle w,r,v\rangle$$

$$A(w,r) := \underline{w}\langle v\rangle.WORK.r(v').[v'=v].A$$

$$\rightarrow$$

$$\underline{w}\langle a\rangle.WORK.r(v').[v'=a].A \mid$$

$$\underline{w}\langle v\rangle.WORK.r(v').[v'=v].A \mid$$

$$w(a).M\langle w,r,f\rangle + \underline{r}\langle v\rangle.M\langle w,r,v\rangle$$

Shared Memory Problems

$$M(w,r,v) := w(f).M<w,r,f> + \underline{r}<v>.M<w,r,v>$$

$$A(w,r) := \underline{w}<v>.WORK.r(v').[v'=v].A$$

→

$$\underline{w}<a>.WORK.r(v').[v'=a].A \mid$$

$$\underline{w}<v>.WORK.r(v').[v'=v].A \mid$$

$$w(a).M<w,r,a> + \underline{r}<v>.M<w,r,v>$$

Shared Memory Problems

$$M(w,r,v) := w(f).M\langle w,r,f\rangle + \underline{r}\langle v\rangle.M\langle w,r,v\rangle$$

$$A(w,r) := \underline{w}\langle v\rangle.\mathrm{WORK}.r(v').[v'=v].A$$

$$\rightarrow$$

$$\underline{w}\langle a\rangle.\mathrm{WORK}.r(v').[v'=a].A \mid$$

$$\underline{w}\langle v\rangle.\mathrm{WORK}.r(v').[v'=v].A \mid$$

$$w(a).M\langle w,r,a\rangle + \underline{r}\langle v\rangle.M\langle w,r,v\rangle$$

Shared Memory Problems

$$M(w,r,v) := w(f).M<w,r,f> + \underline{r}<v>.M<w,r,v>$$

$$A(w,r) := \underline{w}<v>.\mathrm{WORK}.r(v').[v'=v].A$$

$$\rightarrow$$

$$\underline{w}<a>.\mathrm{WORK}.r(v').[v'=a].A \mid$$

$$\underline{w}<v>.\mathrm{WORK}.r(v').[v'=v].A \mid$$

$$w(f).M<w,r,f> + \underline{r}<a>.M<w,r,a>$$

Shared Memory Problems

$M(w,r,v) := w(f).M<w,r,f> + \underline{r}<v>.M<w,r,v>$

$A(w,r) := \underline{w}<v>.WORK.r(v').[v'=v].A$

→

$\underline{w}<a>.WORK.r(v').[v'=a].A \mid$

$\underline{w}<b>.WORK.r(v').[v'=b].A \mid$

$w(b).M<w,r,b> + \underline{r}<a>.M<w,r,a>$

Shared Memory Problems

$M(w,r,v) := w(f).M\langle w,r,f\rangle + \underline{r}\langle v\rangle.M\langle w,r,v\rangle$

$A(w,r) := \underline{w}\langle v\rangle.WORK.r(v').[v'=v].A$

$\rightarrow$

$\underline{w}\langle a\rangle.WORK.r(v').[v'=a].A \mid$

$\underline{w}\langle b\rangle.WORK.r(v').[v'=b].A \mid$

$w(b).M\langle w,r,b\rangle + \underline{r}\langle a\rangle.M\langle w,r,a\rangle$

Shared Memory Problems

M(w,r,v) := w(f).M<w,r,f> + r<v>.M<w,r,v>

A(w,r) := w<v>.WORK.r(v').[v'=v].A

→

w<a>.WORK.r(v').[v'=a].A |

w<b>.WORK.r(v').[v'=b].A |

w(f).M<w,r,f> + r<b>.M<w,r,b>

Shared Memory Problems

$$M(w,r,v) := w(f).M<w,r,f> + \underline{r}<v>.M<w,r,v>$$

$$A(w,r) := \underline{w}<v>.WORK.r(v').[v'=v].A$$

$$\rightarrow$$

$$\underline{w}<a>.WORK.r(v').[b=a].A \mid$$

$$\underline{w}<b>.WORK.r(v').[v'=b].A \mid$$

$$w(f).M<w,r,f> + \underline{r}<b>.M<w,r,b>$$

Shared Memory Problems

M(w,r,v) := w(f).M<w,r,f> + r<v>.M<w,r,v>

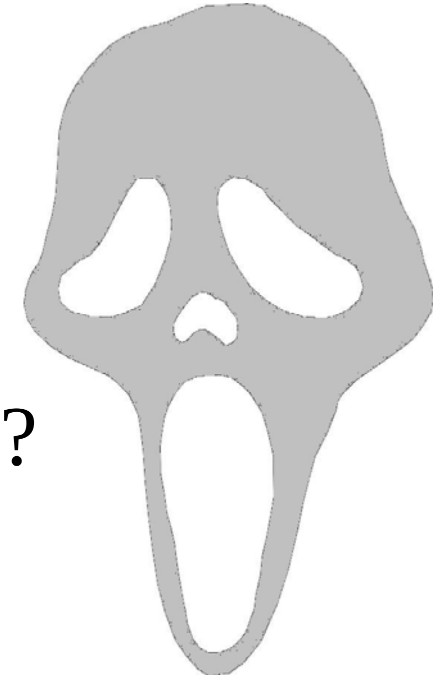
A(w,r) := w<v>.WORK.r(v').[v'=v].A

→

w<a>.WORK.r(v').[b=a].A |

w<b>.WORK.r(v').[v'=b].A |

w(f).M<w,r,f> + r<b>.M<w,r,b>



Locking is dangerous

- What if a thread fails whilst holding a lock?
- Deadlocks happen!
- Linux pros on http://lwn.net/Articles/86859/

Locking does not scale well

Fine grained locked data structures implementation exist of the shelf but…

Global knowlegde!

Transaction

Atomicity

Consistency

Isolation

Durability

Transaction

A + B + C should remain constant under the execution of both transactions in any order.

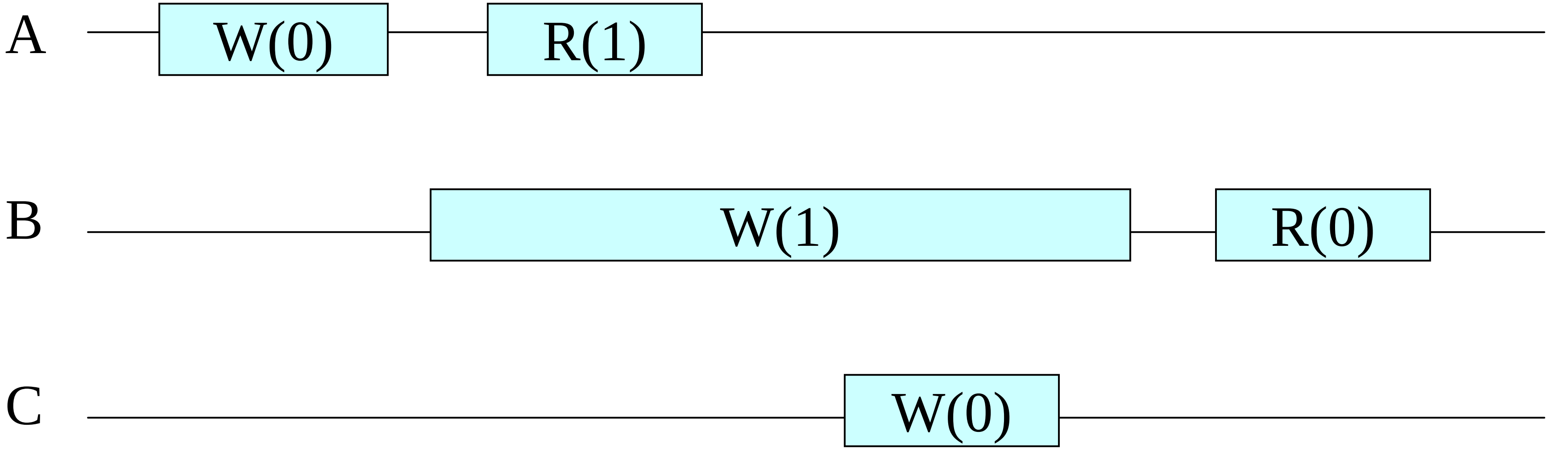
Correct:

$A = A - 10$	lock L_1
lock L_1	$B = B - 20$
$B = B + 10$	unlock L_1
unlock L_1	$C = C + 20$

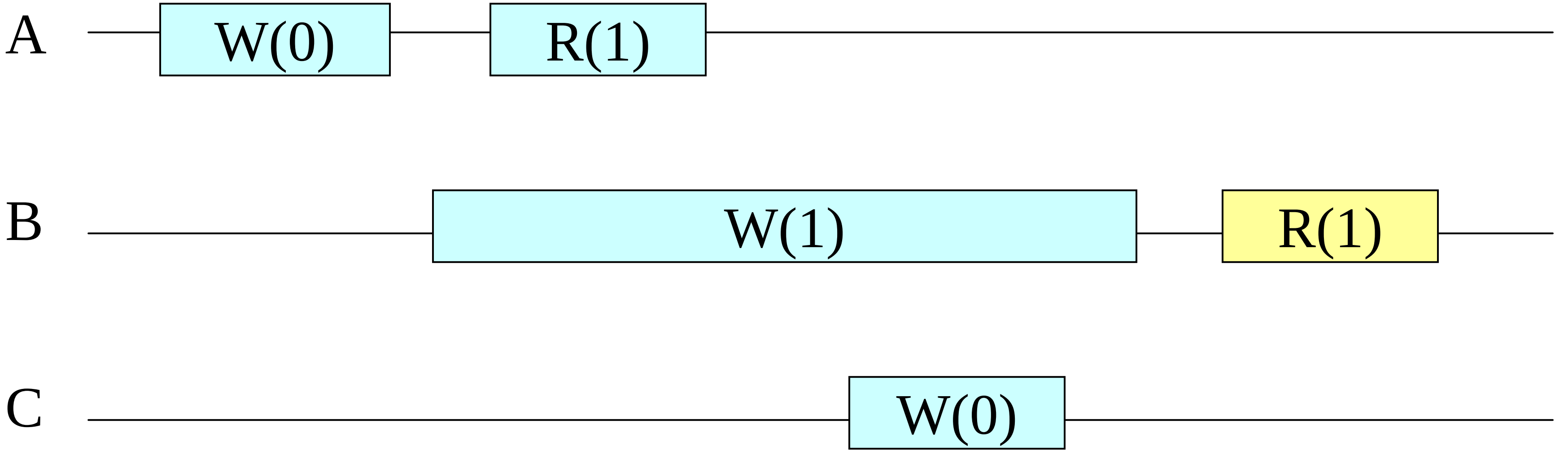
Wrong:

lock L_1	lock L_2
$A = A - 10$	$B = B - 20$
unlock L_1	unlock L_2
$B = B + 10$	$C = C + 20$

Transactional Memory

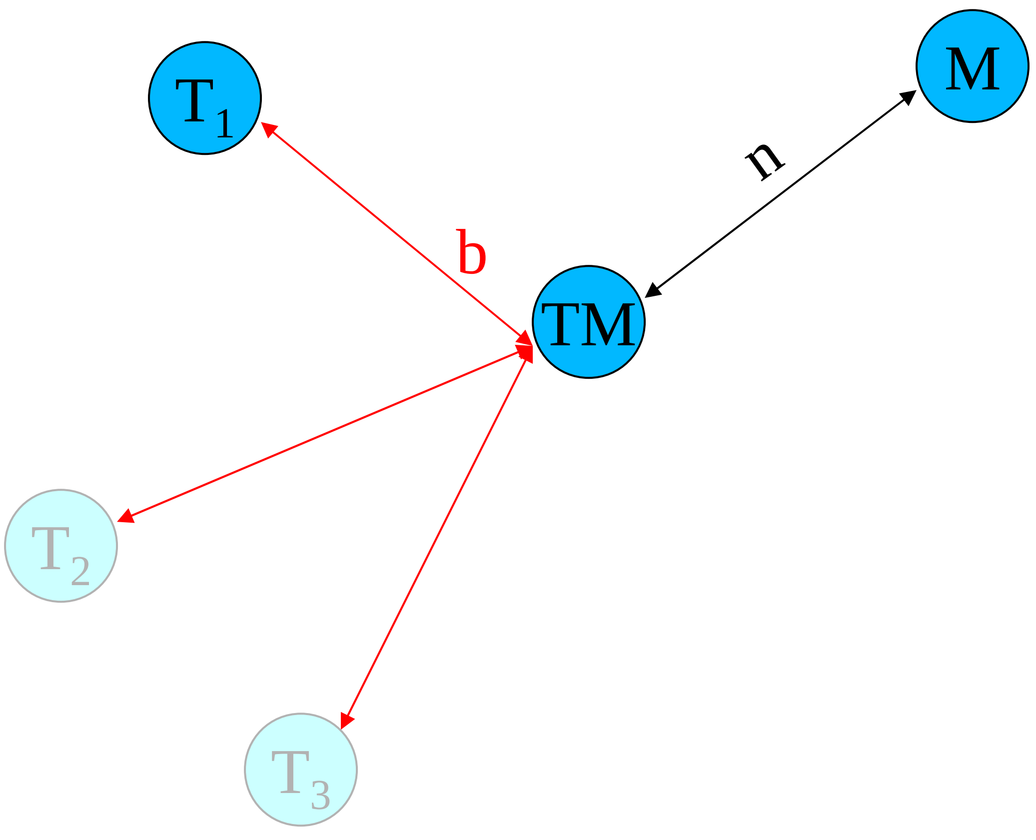


Transactional Memory



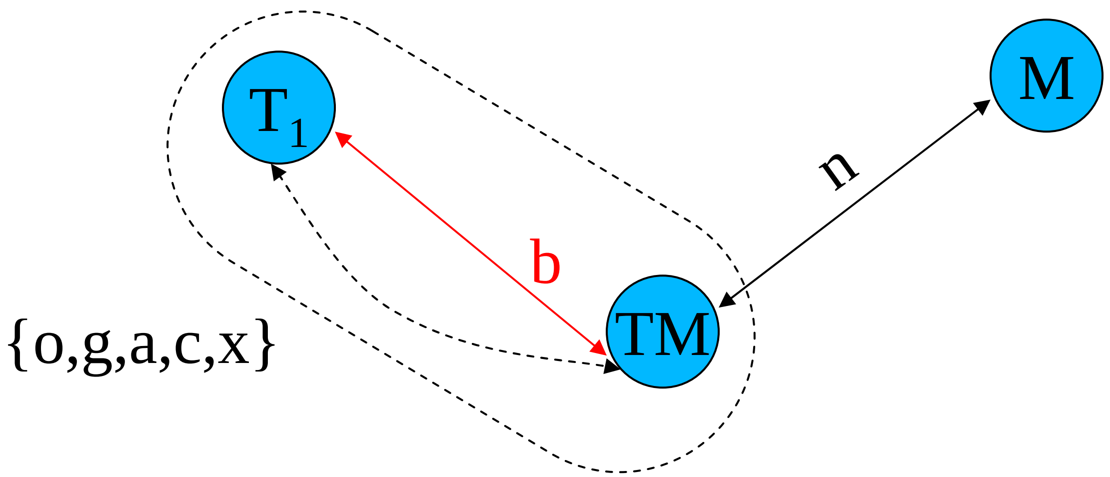
Transactional Memory

Initial Situation:



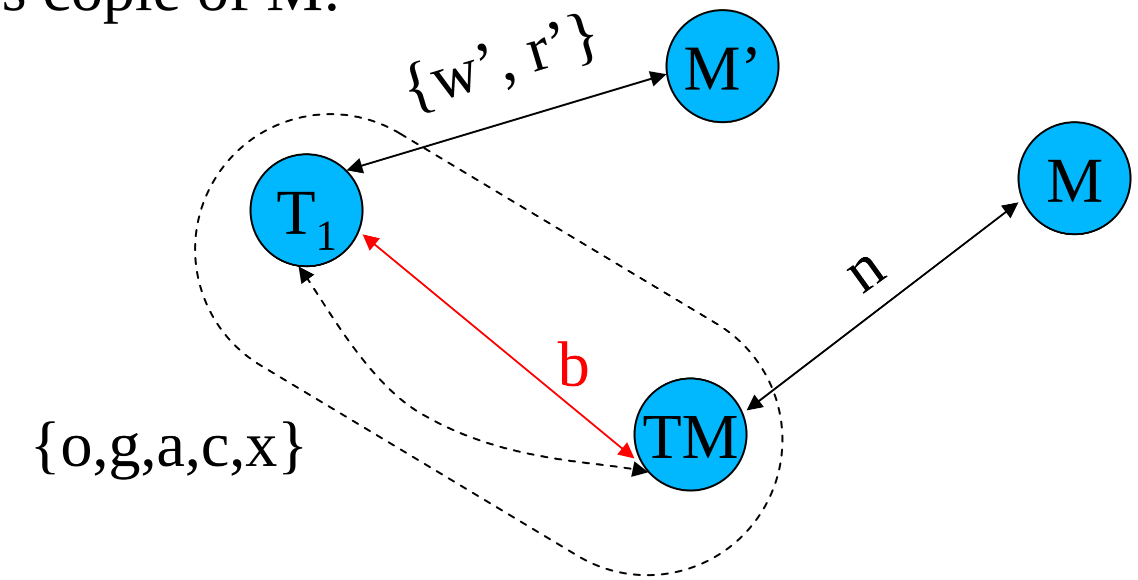
Transactional Memory

T_1 gives his chanels to TM:



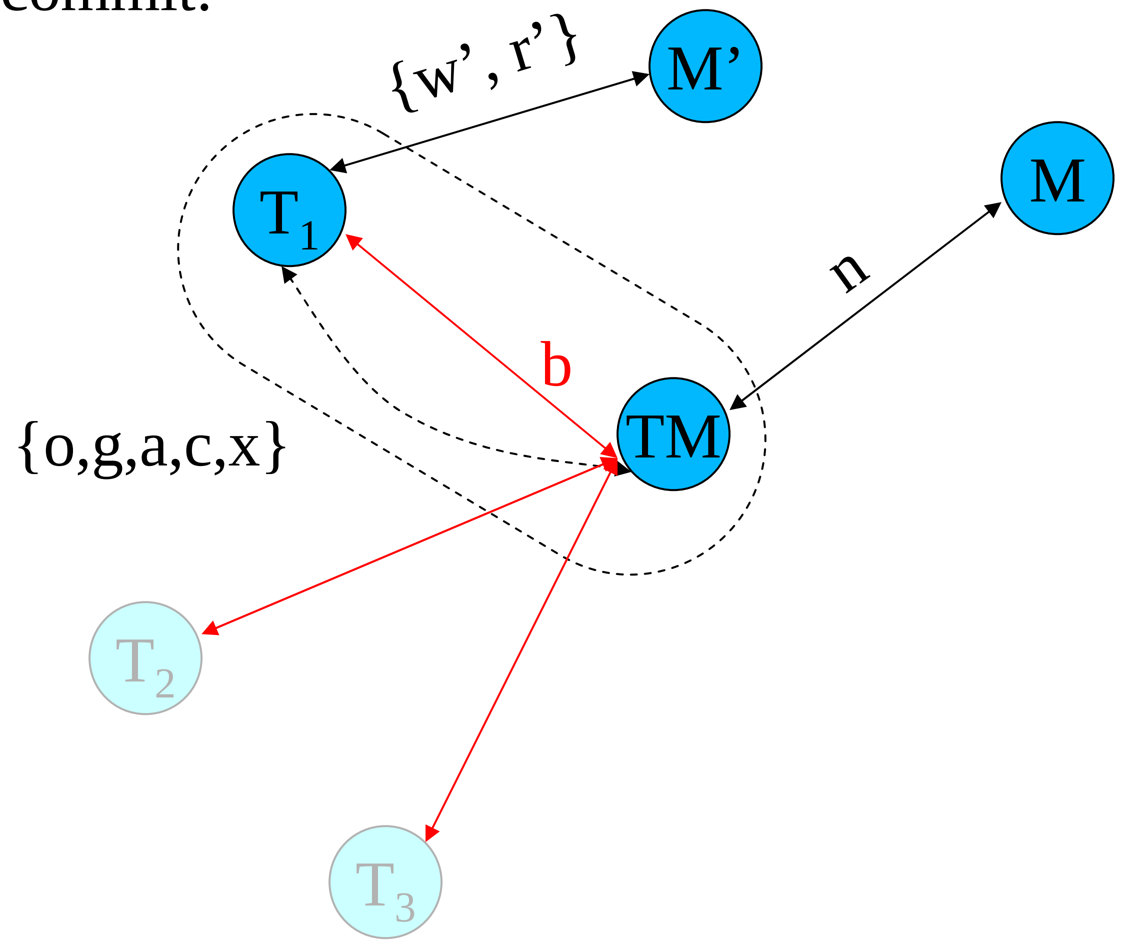
Transactional Memory

T_1 gets it's copie of M:



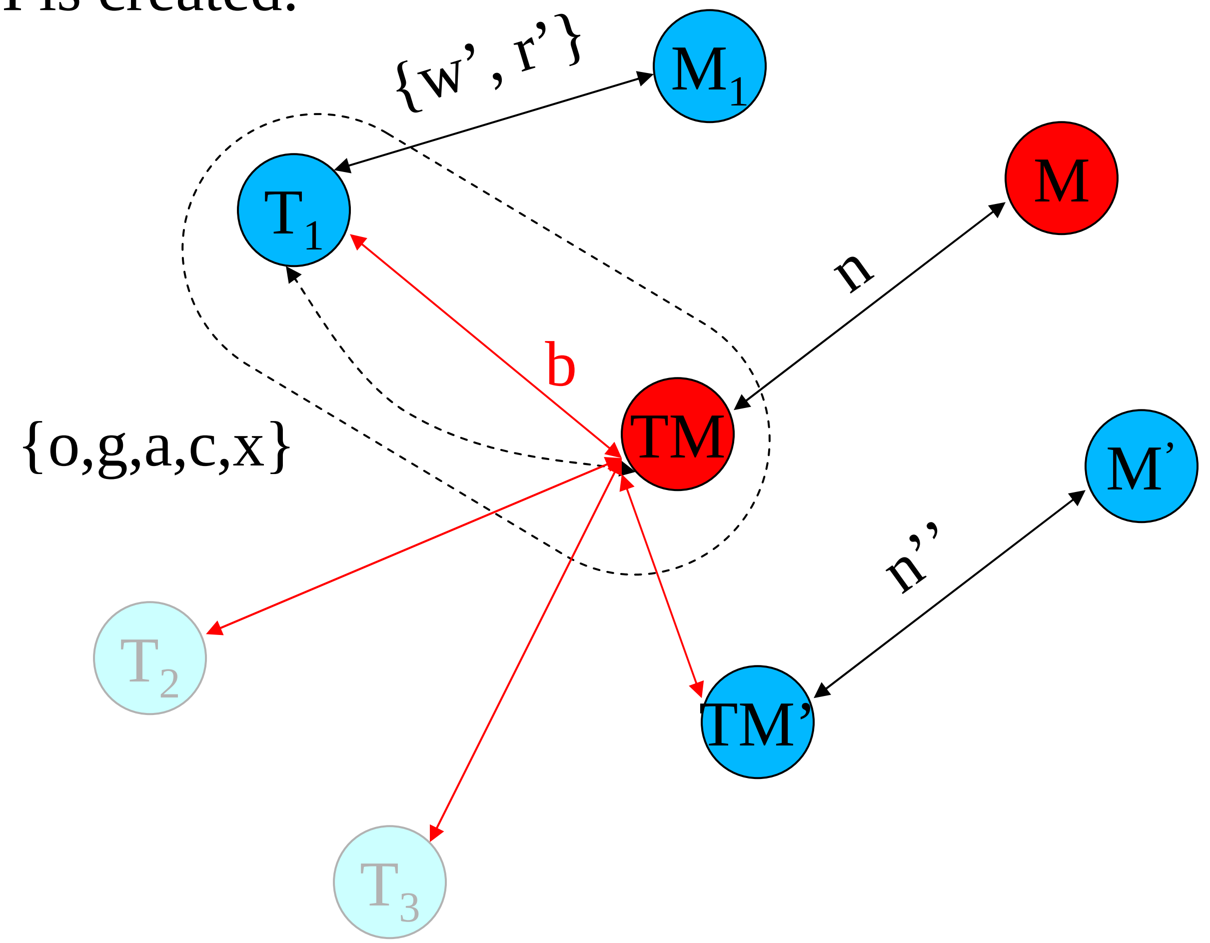
Transactional Memory

T_1 tries to commit:



Transactional Memory

A new TM is created:



Concurrent Commit

$T_1 : \underline{c}_1.(c_1 + x_1.T_1<b>)$

$T_2 : \underline{c}_2.(c_2 + x_2.T_2<b>)$

$TM : \underline{crash} \mid !\ldots \mid k.\underline{x}_1 \mid k.\underline{x}_2 \mid !\; p(x).k.\underline{x} \mid$

$\quad c_1.crash.n_1'(w'',r'',n'').(TM<b,n''> \mid !\underline{k} \mid \underline{c}_1) \mid$

$\quad c_2.crash.n_2'(w'',r'',n'').(TM<b,n''> \mid !\underline{k} \mid \underline{c}_2)$

Concurrent Commit

$T_1 :\ \underline{c}_1.(c_1 + x_1.T_1\langle b\rangle)$

$T_2 :\ \underline{c}_2.(c_2 + x_2.T_2\langle b\rangle)$

$TM : \underline{crash} \mid !\ldots \mid k.\underline{x}_1 \mid k.\underline{x}_2 \mid !\ p(x).k.\underline{x} \mid$

$c_1.crash.n_1'(w'',r'',n'').(TM\langle b,n''\rangle \mid !\underline{k} \mid \underline{c}_1) \mid$

$c_2.crash.n_2'(w'',r'',n'').(TM\langle b,n''\rangle \mid !\underline{k} \mid \underline{c}_2)$

Concurrent Commit

T_1 : $\underline{c}_1.(c_1 + x_1.T_1\langle b\rangle)$

T_2 : $\underline{c}_2.(c_2 + x_2.T_2\langle b\rangle)$

TM : $\underline{\text{crash}}$ | !... | $k.\underline{x}_1$ | $k.\underline{x}_2$ | ! $p(x).k.\underline{x}$ |

$c_1.\text{crash}.n_1'(w'',r'',n'').(TM\langle b,n''\rangle \mid\ !\underline{k} \mid \underline{c}_1)$ |

$c_2.\text{crash}.n_2'(w'',r'',n'').(TM\langle b,n''\rangle \mid\ !\underline{k} \mid \underline{c}_2)$

Concurrent Commit

$T_1 : \; \underline{c}_1.(c_1 + x_1.T_1\langle b\rangle)$

$T_2 : \; \underline{c}_2.(c_2 + x_2.T_2\langle b\rangle)$

$TM : \underline{crash} \mid !\ldots \mid k.\underline{x}_1 \mid k.\underline{x}_2 \mid !\, p(x).k.\underline{x} \mid$

$\quad c_1.crash.n_1'(w'',r'',n'').(TM\langle b,n''\rangle \mid !\underline{k} \mid \underline{c}_1) \mid$

$\quad c_2.crash.n_2'(w'',r'',n'').(TM\langle b,n''\rangle \mid !\underline{k} \mid \underline{c}_2)$

Concurrent Commit

$T_1: \ \underline{c}_1.(c_1 + x_1.T_1\langle b\rangle)$

$T_2: \ \underline{c}_2.(c_2 + x_2.T_2\langle b\rangle)$

$TM: \underline{crash} \mid !\ldots \mid k.\underline{x}_1 \mid k.\underline{x}_2 \mid !\ p(x).k.\underline{x} \mid$

$c_1.crash.n_1'(w'',r'',n'').(TM\langle b,n''\rangle \mid !\underline{k} \mid \underline{c}_1) \mid$

$c_2.crash.n_2'(w'',r'',n'').(TM\langle b,n''\rangle \mid !\underline{k} \mid \underline{c}_2)$

Concurrent Commit

$T_1: \ \underline{c}_1.(c_1 + x_1.T_1<b>)$

$T_2: \ \underline{c}_2.(c_2 + x_2.T_2<b>)$

$TM: \underline{crash} \mid !\ldots \mid k.\underline{x}_1 \mid k.\underline{x}_2 \mid !\, p(x).k.\underline{x} \mid$

$\quad c_1.crash.n_1'(w'',r'',n'').(TM<b,n''> \mid !\underline{k} \mid \underline{c}_1) \mid$

$\quad c_2.crash.n_2'(w'',r'',n'').(TM<b,n''> \mid !\underline{k} \mid \underline{c}_2)$

Concurrent Commit

$$T_1 : \ \underline{c}_1.(c_1 + x_1.T_1\langle b\rangle)$$

$$T_2 : \ \underline{c}_2.(c_2 + x_2.T_2\langle b\rangle)$$

$$\begin{aligned} TM : \ & \underline{\text{crash}} \mid \,!\ldots \mid k.\underline{x}_1 \mid k.\underline{x}_2 \mid \,!\, p(x).k.\underline{x} \mid \\ & c_1.\text{crash}.n_1'(w'',r'',n'').(TM\langle b,n''\rangle \mid \,!\underline{k} \mid \underline{c}_1) \mid \\ & c_2.\text{crash}.n_2'(w'',r'',n'').(TM\langle b,n''\rangle \mid \,!\underline{k} \mid \underline{c}_2) \end{aligned}$$

Concurrent Commit

$T_1: \underline{c}_1.(c_1 + x_1.T_1\langle b\rangle)$

$T_2: \underline{c}_2.(c_2 + x_2.T_2\langle b\rangle)$

$TM : \underline{crash} \mid !\ldots \mid k.\underline{x}_1 \mid k.\underline{x}_2 \mid !\, p(x).k.\underline{x} \mid$

$c_1.crash.n_1'(w'',r'',n'').(TM\langle b,n''\rangle \mid !\underline{k} \mid \underline{c}_1) \mid$

$c_2.crash.n_2'(w'',r'',n'').(TM\langle b,n''\rangle \mid !\underline{k} \mid \underline{c}_2)$

Exceptions Enhancement

- Avoiding Unnecessary Exceptions

TM := (… | !p(x,cancel).(k.$\underline{x}$ + cancel))

G := … (*v*cancel)($\underline{p}$<x,cancel>. … .(… | cancel.!k)

- Allowing Needed Exceptions

Commit while Starting

$T_1 : \ \underline{c}_1.(c_1 + x_1.T_1\langle b\rangle)$

$T_2 : \ \underline{b}\langle o_2,g_2,a_2,c_2,x_2\rangle.\underline{o}.g \ \ldots$

$TM : \underline{crash} \mid !\ldots \mid k.\underline{x}_1 \mid !\ p(x).k.\underline{x} \mid$

$\quad c_1.crash.n_1'(w'',r'',n'').(TM\langle b,n''\rangle \mid !\underline{k} \mid \underline{c}_1) \mid$

$\quad b(o_2,g_2,a_2,c_2,x_2).p\langle x_2\rangle.o \ \ldots$

Commit while Starting

$T_1 :\ \underline{c}_1.(c_1 + x_1.T_1\langle b\rangle)$

$T_2 :\ \underline{b}\langle o_2,g_2,a_2,c_2,x_2\rangle.\underline{o}.g \ldots$

$TM : \underline{crash} \mid !\ldots \mid k.\underline{x}_1 \mid !\ p(x).k.\underline{x} \mid$

$c_1.crash.n_1'(w'',r'',n'').(TM\langle b,n''\rangle \mid !\underline{k} \mid \underline{c}_1) \mid$

$b(o_2,g_2,a_2,c_2,x_2).p\langle x_2\rangle.o \ldots$

Commit while Starting

T_1 : $\underline{c}_1.(c_1 + x_1.T_1\langle b\rangle)$

T_2 : $\underline{b}\langle o_2,g_2,a_2,c_2,x_2\rangle.\underline{o}.g \ldots$

TM : $\underline{\text{crash}} \mid !\ldots \mid k.\underline{x}_1 \mid !\ p(x).k.\underline{x} \mid$

$c_1.\text{crash}.n_1'(w'',r'',n'').(TM\langle b,n''\rangle \mid !\underline{k} \mid \underline{c}_1) \mid$

$b(o_2,g_2,a_2,c_2,x_2).p\langle x_2\rangle.o \ldots$

Commit while Starting

$T_1 : \underline{c}_1.(c_1 + x_1.T_1\langle b\rangle)$

$T_2 : \underline{b}\langle o_2,g_2,a_2,c_2,x_2\rangle.\underline{o}.g \ldots$

$TM : \underline{crash} \mid !\ldots \mid k.\underline{x}_1 \mid !\, p(x).k.\underline{x} \mid$

$c_1.crash.n_1'(w'',r'',n'').(TM\langle b,n''\rangle \mid !\underline{k} \mid \underline{c}_1) \mid$

$b(o_2,g_2,a_2,c_2,x_2).p\langle x_2\rangle.o \ldots$

Commit while Starting

$T_1 : \; \underline{c}_1.(c_1 + x_1.T_1\langle b\rangle)$

$T_2 : \; \underline{b}\langle o_2,g_2,a_2,c_2,x_2\rangle.\underline{o}.g \ldots$

$TM : \underline{crash} \mid !\ldots \mid k.\underline{x}_1 \mid !\, p(x).k.\underline{x} \mid$

$c_1.crash.n_1'(w'',r'',n'').(TM\langle b,n''\rangle \mid !\underline{k} \mid \underline{c}_1) \mid$

$b(o_2,g_2,a_2,c_2,x_2).p\langle x_2\rangle.o \ldots$

Commit while Starting

T_1 : $\underline{c}_1.(c_1 + x_1.T_1\langle b\rangle)$

T_2 : $\underline{b}\langle o_2,g_2,a_2,c_2,x_2\rangle.\underline{o}.g$ …

TM : $\underline{crash}$ | !... | $k.\underline{x}_1$ | ! $p(x).k.\underline{x}$ |

$c_1.crash.n_1'(w'',r'',n'').(TM\langle b,n''\rangle \mid !\underline{k} \mid \underline{c}_1)$ |

$b(o_2,g_2,a_2,c_2,x_2).p\langle x_2\rangle.o$ …

Commit while Starting

T_1 : $\underline{c}_1.(c_1 + x_1.T_1<b>)$

T_2 : $\underline{b}<o_2,g_2,a_2,c_2,x_2>.\underline{o}.g \ldots$

TM : $\underline{crash} \mid !\ldots \mid k.\underline{x}_1 \mid p(x).k.\underline{x} \mid !\, p(x).k.\underline{x} \mid$

$c_1.crash.n_1'(w'',r'',n'').(TM<b,n''> \mid !\underline{k} \mid \underline{c}_1) \mid$

$b(o_2,g_2,a_2,c_2,x_2).p<x_2>.o \ldots$

Commit while Starting

$$T_1 : \ \underline{c}_1.(c_1 + x_1.T_1\langle b\rangle)$$

$$T_2 : \ \underline{b}\langle o_2,g_2,a_2,c_2,x_2\rangle.\underline{o}.g \ \ldots$$

$$TM : \underline{\mathrm{crash}} \mid !\ldots \mid k.\underline{x}_1 \mid p(x).k.\underline{x}_2 \mid !\ p(x).k.\underline{x} \mid$$

$$c_1.\mathrm{crash}.n_1'(w'',r'',n'').(TM\langle b,n''\rangle \mid !\underline{k} \mid \underline{c}_1) \mid$$

$$b(o_2,g_2,a_2,c_2,x_2).p\langle x_2\rangle.o \ \ldots$$

Commit while Starting

$T_1 : \; \underline{c}_1.(c_1 + x_1.T_1<b>)$

$T_2 : \; \underline{b}<o_2,g_2,a_2,c_2,x_2>.\underline{o}.g \ldots$

$TM : \underline{crash} \mid !\ldots \mid k.\underline{x}_1 \mid p(x).k.\underline{x}_2 \mid !\, p(x).k.\underline{x} \mid$

$c_1.crash.n_1'(w'',r'',n'').(TM<b,n''> \mid !\underline{k} \mid \underline{c}_1) \mid$

$b(o_2,g_2,a_2,c_2,x_2).p<x_2>.o \ldots$