Reasoning about correctness of software transactional memory

Andri Saar Tarmo Uustalu

Institute of Cybernetics at
Tallinn University of Technology

ROSAEC seminar, Seoul, 14 Feb. 2011

Motivation

- Databases have inspired software transactional memory (software transactions) as an appealing paradigm of concurrency, an alternative to critical sections
- But how to specify an implementation of transactional memory at a level amenable for mathematical reasoning?
- When is it correct?
- In literature, we encounter multiple correctness criteria, of different level of formality, some too vague to formalize at all.

This talk

- We formalize Guerraoui and Kapałka's opacity using small-step operational semantics and show a transactional memory implementation to meet it.

Language

Expressions and statements:

$$
\begin{aligned}
a ::= \;& a_1 + a_2 \mid a_1 - a_2 \mid a_1 * a_2 \\
b ::= \;& a_1 < a_2 \mid a_1 = a_2 \mid \neg b \mid b_1 \vee b_2 \mid b_1 \wedge b_2 \\
s ::= \;& x := a \mid \texttt{if } b \texttt{ then } s_1 \texttt{ else } s_2 \mid \texttt{while } b \texttt{ do } s \\
\mid \;& \texttt{skip} \mid s_1; s_2 \\
\mid \;& s_1 \parallel s_2 \\
\mid \;& \texttt{trans } s
\end{aligned}
$$

Transactional semantics ("weak model"): Extended statements

Write, read caches wc, rc: partial mappings from variable names to values

Extended statements (used in the semantics only):

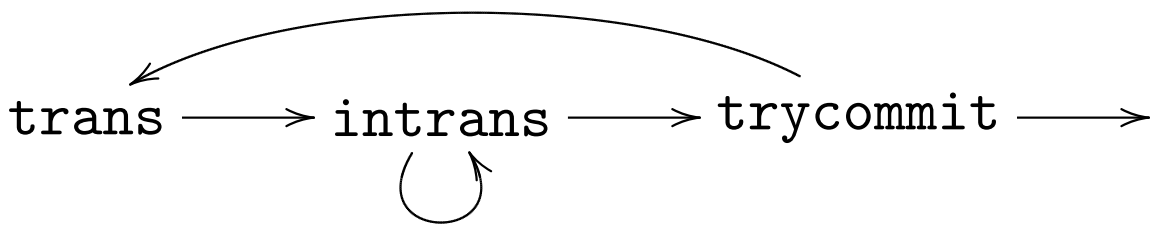
$$xs ::= x := a \mid \texttt{if } b \texttt{ then } s_1 \texttt{ else } s_2 \mid \texttt{while } b \texttt{ do } s \mid \texttt{skip}$$
$$\mid xs; s \mid xs_1 \parallel xs_2 \mid \texttt{trans } s$$
$$\mid \texttt{intrans } xs\ s\ wc\ rc \mid \texttt{trycommit } s\ wc\ rc$$

Transactional semantics ("weak model"): Extended statements

Write, read caches *wc*, *rc*: partial mappings from variable names to values

Extended statements (used in the semantics only):

$$
\begin{aligned}
xs ::= {} & x := a \mid \texttt{if } b \texttt{ then } s_1 \texttt{ else } s_2 \mid \texttt{while } b \texttt{ do } s \mid \texttt{skip} \\
& \mid xs; s \mid xs_1 \parallel xs_2 \mid \texttt{trans } s \\
& \mid \texttt{intrans } xs\ s\ wc\ rc \mid \texttt{trycommit } s\ wc\ rc
\end{aligned}
$$



Transactional semantics: Extended memories and configurations

Memories m: total mappings from variable names to values

Extended memories (write, read caches of parent transactions + main memory):

$$xm ::= m \mid wc : rc : xm$$

Extended configurations:

$$xcfg ::= \langle xm \rangle \mid \langle xs, xm \rangle$$

Operations on extended memories

$$\mathsf{read}(x, wc : rc : xm) = \begin{cases} wc\ x & \text{if } x \in \mathsf{dom}\ wc \\ rc\ x & \text{if } x \notin \mathsf{dom}\ wc \wedge x \in \mathsf{dom}\ rc \\ \mathsf{read}(x, xm) & \text{otherwise} \end{cases}$$

$$\mathsf{read}(x, m) = m\ x$$

Operations on extended memories

$$\mathrm{read}(x, wc : rc : xm) = \begin{cases} wc\ x & \text{if } x \in \mathrm{dom}\ wc \\ rc\ x & \text{if } x \notin \mathrm{dom}\ wc \wedge x \in \mathrm{dom}\ rc \\ \mathrm{read}(x, xm) & \text{otherwise} \end{cases}$$

$$\mathrm{read}(x, m) = m\ x$$

$$\mathrm{write}\,(c, wc : rc : xm) = \left[\begin{array}{ll} c\ x & \text{if } x \in \mathrm{dom}\ c \\ wc\ x & \text{otherwise} \end{array} \middle| x \right] : rc : xm$$

$$\mathrm{write}\,(c, m) = \left[\begin{array}{ll} c\ x & \text{if } x \in \mathrm{dom}\ c \\ m\ x & \text{otherwise} \end{array} \middle| x \right]$$

Operations on extended memories

$$\mathrm{read}(x, wc : rc : xm) = \begin{cases} wc\ x & \text{if } x \in \mathrm{dom}\ wc \\ rc\ x & \text{if } x \notin \mathrm{dom}\ wc \wedge x \in \mathrm{dom}\ rc \\ \mathrm{read}(x, xm) & \text{otherwise} \end{cases}$$

$$\mathrm{read}(x, m) = m\ x$$

$$\mathrm{write}\,(c, wc : rc : xm) = \left[\begin{array}{ll} c\ x & \text{if } x \in \mathrm{dom}\ c \\ wc\ x & \text{otherwise} \end{array} \middle| x \right] : rc : xm$$

$$\mathrm{write}\,(c, m) = \left[\begin{array}{ll} c\ x & \text{if } x \in \mathrm{dom}\ c \\ m\ x & \text{otherwise} \end{array} \middle| x \right]$$

$$\mathrm{rcupdate}(c, wc : rc : xm) = wc : \left[\begin{array}{ll} c\ x & \text{if } x \in \mathrm{dom}\ c \wedge \\ & \quad x \notin \mathrm{dom}\ rc \wedge \\ & \quad x \notin \mathrm{dom}\ wc \\ rc\ x & \text{otherwise} \end{array} \middle| x \right] : xm$$

$$\mathrm{rcupdate}(c, m) = m$$

Consistency

- An extended memory xm is *consistent* with a cache c (written $xm \approx c$) iff
  $\forall x \in \mathrm{dom}\, c\ \mathrm{read}(x, xm) = c\ x$.
- An extended memory xm is *consistent* with a memory m (written $xm \approx m$) iff
  $\forall x\ \mathrm{read}(x, xm) = m\ x$.

Consistency

- An extended memory xm is *consistent* with a cache c (written $xm \approx c$) iff
  $\forall x \in \mathrm{dom}\, c\ \mathrm{read}(x, xm) = c\ x$.
- An extended memory xm is *consistent* with a memory m (written $xm \approx m$) iff
  $\forall x\ \mathrm{read}(x, xm) = m\ x$.
- For any extended memory xm,
  $\forall x\ \mathrm{read}(x, [] : [] : xm) = \mathrm{read}(x, xm)$

Consistency

- An extended memory xm is *consistent* with a cache c (written $xm \approx c$) iff
  $\forall x \in \mathrm{dom}\, c\ \mathrm{read}(x, xm) = c\ x$.
- An extended memory xm is *consistent* with a memory m (written $xm \approx m$) iff
  $\forall x\ \mathrm{read}(x, xm) = m\ x$.
- For any extended memory xm,
  $\forall x\ \mathrm{read}(x, [] : [] : xm) = \mathrm{read}(x, xm)$
- For any caches wc, rc
  and extended memory xm consistent with rc,
  $\forall x\ \mathrm{read}(x, wc : rc : xm) =$
  $\mathrm{read}(x, \mathrm{write}\,(wc, \mathrm{rcupdate}(rc, xm)))$.

Transactional semantics: Reduction rules (excerpt)

$$\frac{\mathsf{read}(x, xm) = v}{\langle x, xm\rangle \Rightarrow_{WA} \langle v, \mathsf{rcupdate}([x \mapsto v], xm)\rangle}$$

$$\frac{\langle a, xm\rangle \Rightarrow_{WA} \langle v, xm'\rangle}{\langle x := a, xm\rangle \rightarrow_{W} \langle \mathsf{write}\,([x \mapsto v], xm')\rangle}$$

$$\frac{}{\langle \texttt{trans}\ s, xm\rangle \rightarrow_{W} \langle \texttt{intrans}\ s\ s\ []\ []\,, xm\rangle}$$

$$\frac{\langle xs, wc : rc : xm\rangle \rightarrow_{W} \langle xs', wc' : rc' : xm\rangle}{\langle \texttt{intrans}\ xs\ s\ wc\ rc, xm\rangle \rightarrow_{W} \langle \texttt{intrans}\ xs'\ s\ wc'\ rc', xm\rangle}$$

$$\frac{\langle xs, wc : rc : xm\rangle \rightarrow_{W} wc' : rc' : xm}{\langle \texttt{intrans}\ xs\ s\ wc\ rc, xm\rangle \rightarrow_{W} \langle \texttt{trycommit}\ s\ wc'\ rc', xm\rangle}$$

$$\frac{xm \approx rc}{\langle \texttt{trycommit}\ s\ wc\ rc, xm\rangle \rightarrow_{W} \langle \mathsf{write}\,(wc, \mathsf{rcupdate}(rc, xm))\rangle}$$

$$\frac{xm \not\approx rc}{\langle \texttt{intrans}\ xs\ s\ wc\ rc, xm\rangle \rightarrow_{W} \langle \texttt{trans}\ s, xm\rangle}$$

$$\frac{xm \not\approx rc}{\langle \texttt{trycommit}\ s\ wc\ rc, xm\rangle \rightarrow_{W} \langle \texttt{trans}\ s, xm\rangle}$$

Opacity

We have to show that for every run in the transactional semantics there exists some equivalent run, that satisfies the following properties:

1. It is *sequential*, that is, contains no two transactions that are concurrent.
2. It contains no live transactions.
3. It preserves the real-time order of the original run.
4. Every transaction in the equivalent run is *legal*.

Sequential semantics ("strong model")

Statements, memories and configurations:

$$
\begin{aligned}
s &::= x := a \mid \texttt{if } b \texttt{ then } s_1 \texttt{ else } s_2 \mid \texttt{while } b \texttt{ do } s \mid \texttt{skip} \\
&\mid s_1; s_2 \mid s_1 \parallel s_2 \mid \texttt{trans } s \\
\mathit{cfg} &::= \langle m \rangle \mid \langle s, m \rangle
\end{aligned}
$$

Sequential semantics ("strong model")

Statements, memories and configurations:

$$s ::= x := a \mid \texttt{if } b \texttt{ then } s_1 \texttt{ else } s_2 \mid \texttt{while } b \texttt{ do } s \mid \texttt{skip}$$
$$\mid s_1; s_2 \mid s_1 \parallel s_2 \mid \texttt{trans } s$$
$$cfg ::= \langle m \rangle \mid \langle s, m \rangle$$

Reduction rules (excerpt):

$$\frac{\langle a, m \rangle \Rightarrow_{SA} v}{\langle x := a, m \rangle \rightarrow_S \langle m[x \mapsto v] \rangle}$$

$$\frac{\langle s, m \rangle \rightarrow_S^* \langle m' \rangle}{\langle \texttt{trans } s, m \rangle \rightarrow_S \langle m' \rangle}$$

$\rightarrow^*$ is the reflexive and transitive closure of $\rightarrow$.

Relating the two semantics (to align runs)

$$\frac{}{\texttt{if } b \texttt{ then } s_1 \texttt{ else } s_2 \sim \texttt{if } b \texttt{ then } s_1 \texttt{ else } s_2} \qquad \frac{}{\texttt{skip} \sim \texttt{skip}}$$

$$\frac{}{\texttt{while } b \texttt{ do } s \sim \texttt{while } b \texttt{ do } s} \qquad \frac{xs \sim s'}{xs; s \sim s'; s} \qquad \frac{xs_1 \sim s_1 \quad xs_2 \sim s_2}{xs_1 \parallel xs_2 \sim s_1 \parallel s_2}$$

$$\frac{}{\texttt{trans } s \sim \texttt{trans } s}$$

$$\frac{\forall xm.\, xm \approx rc \;\Rightarrow\; \langle s, [] : [] : xm\rangle \rightarrow_W^* \langle xs, wc : rc : xm\rangle}{\texttt{intrans } xs\ s\ wc\ rc \sim \texttt{trans } s}$$

$$\frac{\forall xm.\, xm \approx rc \;\Rightarrow\; \langle s, [] : [] : xm\rangle \rightarrow_W^* \langle wc : rc : xm\rangle}{\texttt{trycommit } s\ \ wc\ rc \sim \texttt{trans } s}$$

Relating the two semantics (to align runs)

$$\frac{}{\texttt{if } b \texttt{ then } s_1 \texttt{ else } s_2 \sim \texttt{if } b \texttt{ then } s_1 \texttt{ else } s_2} \qquad \frac{}{\texttt{skip} \sim \texttt{skip}}$$

$$\frac{}{\texttt{while } b \texttt{ do } s \sim \texttt{while } b \texttt{ do } s} \qquad \frac{xs \sim s'}{xs; s \sim s'; s} \qquad \frac{xs_1 \sim s_1 \quad xs_2 \sim s_2}{xs_1 \parallel xs_2 \sim s_1 \parallel s_2}$$

$$\frac{}{\texttt{trans } s \sim \texttt{trans } s}$$

$$\frac{\forall xm.\, xm \approx rc \;\Rightarrow\; \langle s, [] : [] : xm\rangle \rightarrow_W^* \langle xs, wc : rc : xm\rangle}{\texttt{intrans } xs\ s\ wc\ rc \sim \texttt{trans } s}$$

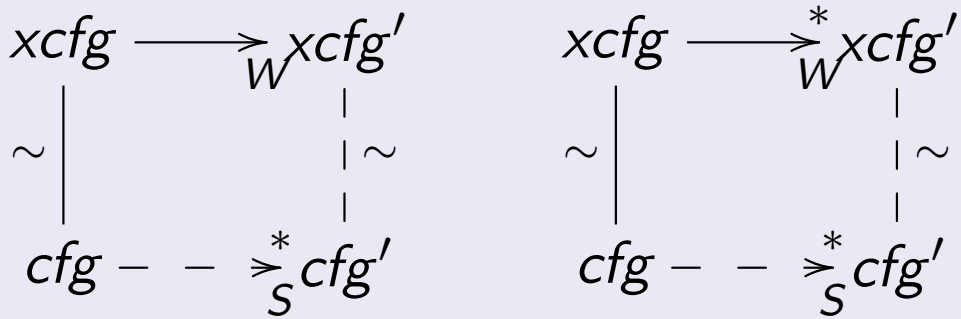
$$\frac{\forall xm.\, xm \approx rc \;\Rightarrow\; \langle s, [] : [] : xm\rangle \rightarrow_W^* \langle wc : rc : xm\rangle}{\texttt{trycommit } s\ \ wc\ rc \sim \texttt{trans } s}$$

$$\frac{xm \approx m}{\langle xm\rangle \sim \langle m\rangle} \qquad \frac{xs \sim s \quad xm \approx m}{\langle xs, xm\rangle \sim \langle s, m\rangle}$$

Correctness argument

Theorem

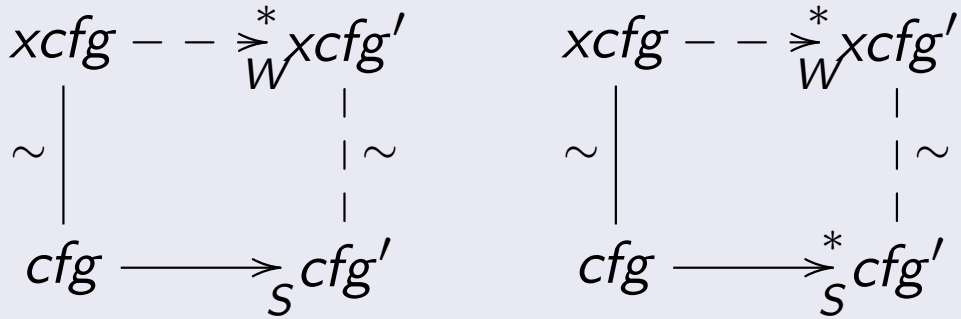
- *For all xcfg, xcfg′ and cfg such that* $xcfg \rightarrow_W xcfg'$ *and* $xcfg \sim cfg$, *there exists cfg′ such that* $cfg \rightarrow_S^* cfg'$ *and* $xcfg' \sim cfg'$.
- *For all xcfg, xcfg′ and cfg such that* $xcfg \rightarrow_W^* xcfg'$ *and* $xcfg \sim cfg$, *there exists cfg′ such that* $cfg \rightarrow_S^* cfg'$ *and* $xcfg' \sim cfg'$.



Correctness argument (2)

Theorem

- *For all cfg, cfg′ and xcfg such that* $cfg \rightarrow_S cfg'$ *and* $xcfg \sim cfg$, *there exists xcfg′ such that* $xcfg \rightarrow_W^* xcfg'$ *and* $xcfg' \sim cfg'$.
- *For all cfg, cfg′ and xcfg such that* $cfg \rightarrow_S^* cfg'$ *and* $xcfg \sim cfg$, *there exists xcfg′ such that* $xcfg \rightarrow_W^* xcfg'$ *and* $xcfg' \sim cfg'$.



Example

Weak model

$\texttt{trans}\ (y := x; x := 2)$

$\|$

$z := 5; \texttt{trans}\ (z := x + z; x := 1)$

$\sim$

$$\begin{bmatrix} x & y & z \\ 0 & 0 & 0 \end{bmatrix}$$

Strong model

$\texttt{trans}\ (y := x; x := 2)$

$\|$

$z := 5; \texttt{trans}\ (z := x + z; x := 1)$

Example

Weak model

$\texttt{intrans}\ (y := x; x := 2)\ (y := x; x := 2) \left[\ \right] \left[\ \right]$

$\|$

$z := 5; \texttt{trans}\ (z := x + z; x := 1)$

$\sim$

$\begin{bmatrix} x & y & z \\ 0 & 0 & 0 \end{bmatrix}$

Strong model

$\texttt{trans}\ (y := x; x := 2)$

$\|$

$z := 5; \texttt{trans}\ (z := x + z; x := 1)$

Example

Weak model

$\texttt{intrans}\ x := 2\ (y := x; x := 2) \begin{bmatrix} y \\ 0 \end{bmatrix} \begin{bmatrix} x \\ 0 \end{bmatrix}$

$\|$

$z := 5; \texttt{trans}\ (z := x + z; x := 1)$

$\sim$

$\begin{bmatrix} x & y & z \\ 0 & 0 & 0 \end{bmatrix}$

Strong model

$\texttt{trans}\ (y := x; x := 2)$

$\|$

$z := 5; \texttt{trans}\ (z := x + z; x := 1)$

Example

Weak model

$\texttt{intrans}\ x := 2\ (y := x; x := 2) \begin{bmatrix} y \\ 0 \end{bmatrix} \begin{bmatrix} x \\ 0 \end{bmatrix}$

$\|$

$\texttt{trans}\ (z := x + z; x := 1)$

$\sim$

$\begin{bmatrix} x & y & z \\ 0 & 0 & 5 \end{bmatrix}$

Strong model

$\texttt{trans}\ (y := x; x := 2)$

$\|$

$\texttt{trans}\ (z := x + z; x := 1)$

Example

Weak model

$\texttt{intrans}\ x := 2\ (y := x; x := 2) \begin{bmatrix} y \\ 0 \end{bmatrix} \begin{bmatrix} x \\ 0 \end{bmatrix}$

$\|$

$\texttt{intrans}\ (z := x + z; x := 1)\ (z := x + z; x := 1) \begin{bmatrix} \\ \end{bmatrix} \begin{bmatrix} \\ \end{bmatrix}$

$\sim$

$\begin{bmatrix} x & y & z \\ 0 & 0 & 5 \end{bmatrix}$

Strong model

$\texttt{trans}\ (y := x; x := 2)$

$\|$

$\texttt{trans}\ (z := x + z; x := 1)$

Example

Weak model

$\texttt{intrans}\ x := 2\ (y := x; x := 2) \begin{bmatrix} y \\ 0 \end{bmatrix} \begin{bmatrix} x \\ 0 \end{bmatrix}$

$\|$

$\texttt{intrans}\ (x := 1)\ (z := x + z; x := 1) \begin{bmatrix} z \\ 5 \end{bmatrix} \begin{bmatrix} x\ z \\ 0\ 0 \end{bmatrix}$

$\sim$

$\begin{bmatrix} x\ y\ z \\ 0\ 0\ 5 \end{bmatrix}$

Strong model

$\texttt{trans}\ (y := x; x := 2)$

$\|$

$\texttt{trans}\ (z := x + z; x := 1)$

Example

Weak model

$\texttt{intrans}\ x := 2\ (y := x; x := 2) \begin{bmatrix} y \\ 0 \end{bmatrix} \begin{bmatrix} x \\ 0 \end{bmatrix}$

$\|$

$\texttt{trycommit}\ (z := x + z; x := 1) \begin{bmatrix} x\ z \\ 1\ 5 \end{bmatrix} \begin{bmatrix} x\ z \\ 0\ 0 \end{bmatrix}$

$\sim$

$\begin{bmatrix} x\ y\ z \\ 0\ 0\ 5 \end{bmatrix}$

Strong model

$\texttt{trans}\ (y := x; x := 2)$

$\|$

$\texttt{trans}\ (z := x + z; x := 1)$

Example

Weak model

$\texttt{trycommit}\ (y := x; x := 2)$ $\begin{bmatrix} x & y \\ 2 & 0 \end{bmatrix}$ $\begin{bmatrix} x \\ 0 \end{bmatrix}$

$\|$

$\texttt{trycommit}\ (z := x + z; x := 1)$ $\begin{bmatrix} x & z \\ 1 & 5 \end{bmatrix}$ $\begin{bmatrix} x & z \\ 0 & 0 \end{bmatrix}$

$\sim$

$\begin{bmatrix} x & y & z \\ 0 & 0 & 5 \end{bmatrix}$

Strong model

$\texttt{trans}\ (y := x; x := 2)$

$\|$

$\texttt{trans}\ (z := x + z; x := 1)$

Example

Weak model

$\texttt{trycommit}\ (z := x + z; x := 1)\ \begin{bmatrix} x & z \\ 1 & 5 \end{bmatrix}\ \begin{bmatrix} x & z \\ 0 & 0 \end{bmatrix}$

$\sim$

$\begin{bmatrix} x & y & z \\ 2 & 0 & 5 \end{bmatrix}$

Strong model

$\texttt{trans}\ (z := x + z; x := 1)$

Example

Weak model

$\texttt{trans}\ (z := x + z; x := 1)$

$\sim$

$$\begin{bmatrix} x & y & z \\ 2 & 0 & 5 \end{bmatrix}$$

Strong model

$\texttt{trans}\ (z := x + z; x := 1)$

Example

Weak model

$\texttt{intrans}\ (z := x + z; x := 1)\ (z := x + z; x := 1) \left[\ \right] \left[\ \right]$

$\sim$

$\begin{bmatrix} x & y & z \\ 2 & 0 & 5 \end{bmatrix}$

Strong model

$\texttt{trans}\ (z := x + z; x := 1)$

Example

Weak model

$$\texttt{intrans}\ x := 1\ (z := x + z; x := 1) \begin{bmatrix} z \\ 7 \end{bmatrix} \begin{bmatrix} x\, z \\ 2\, 5 \end{bmatrix}$$

$\sim$

$$\begin{bmatrix} x\, y\, z \\ 2\, 0\, 5 \end{bmatrix}$$

Strong model

$$\texttt{trans}\ (z := x + z; x := 1)$$

Example

Weak model

$\texttt{trycommit}\ (z := x + z; x := 1)\ \begin{bmatrix} x & z \\ 1 & 7 \end{bmatrix}\ \begin{bmatrix} x & z \\ 2 & 5 \end{bmatrix}$

$\sim$

$\begin{bmatrix} x & y & z \\ 2 & 0 & 5 \end{bmatrix}$

Strong model

$\texttt{trans}\ (z := x + z; x := 1)$

Example

Weak model

$$\begin{bmatrix} x & y & z \\ 1 & 0 & 7 \end{bmatrix}$$

$\sim$

$$\begin{bmatrix} x & y & z \\ 1 & 0 & 7 \end{bmatrix}$$

Strong model

$$\begin{bmatrix} x & y & z \\ 1 & 0 & 7 \end{bmatrix}$$

Conclusion

- Transactional memory simple on the level of the idea, particular implementations subtle
- It is difficult to state mathematical criteria of correctness
- Opacity can be mathematized in terms of aligning runs in two semantics
- Formalization in Agda
- A program logic for the transactional semantics?
- Translate program proofs for the sequential semantics to proofs for the transactional semantics?
- Carry out the same project for other implementations of transactional memory (eg with eager conflict detection)
- Do similar techniques apply to weak memory models?