Compositional Analysis of Resource Bounds for Software Transactions

Mai Thuong Tran, Martin Steffen, and Hoang Truong

University of Oslo, Norway
Vietnam National University, Việt Nam

NWPT 2012,
Bergen, Norway, 31.10 – 2.11 2012
Motivation

- software transactions: modern concurrency control mechanism
- proposed/being developed for a number of PLs
- enhanced performance + programmability
- price to pay: memory resource consumption
optimistic concurrency control: not “prevent” potential interference at the entry of a CR, but check and potentially repair/compensate/undo (potential) conflicts at the end

conflict management (conflict detection + potential roll-back) ⇒ info to reconstruct the original state needs to be stored.
TFJ: formal proposal for Java + transactions
[Jagannathan et al., 2005]

transactions model:
- nested
- multi-threaded
- non-lexical scope

“inheritance” of the resource consumption of parent thread

child threads: joining commit ⇒ implicit synchronization ⇒ main complication
TFJ syntax

\[
P ::= 0 \mid P \parallel P \mid p(e)\]

\[
L ::= \text{class } C\{\vec{f}:\vec{T}; K; \vec{M}\}
\]

\[
K ::= C(\vec{f}:\vec{T})\{\text{this.}\vec{f} := \vec{f}\}
\]

\[
M ::= m(\vec{x}:\vec{T})\{e\} : T
\]

\[
e ::= v \mid v.f \mid v.f := v \mid \text{if } v \text{ then } e \text{ else } e
\]

\[
\mid \text{let } x:T = e \text{ in } e \mid v.m(\vec{v})
\]

\[
\mid \text{new } C(\vec{v}) \mid \text{spawn } e \mid \text{onacid} \mid \text{commit}
\]

\[
v ::= r \mid x \mid \text{null}
\]

processes/threads

class definitions

constructors

methods

expressions

values
Nested and multi-threaded transactions
Nested and multi-threaded transactions
Nested and multi-threaded transactions
Nested and multi-threaded transactions

$l_1: log_1$

$l_1$

$l_2$

$l_3$
Nested and multi-threaded transactions
Nested and multi-threaded transactions

$l_1: \log_2$
Nested and multi-threaded transactions

$l_1: \log_2, l_2: \emptyset$
Nested and multi-threaded transactions

$l_1: \log_2, l_2: \log_3$
Nested and multi-threaded transactions

\[ l_1 : \log_2 \]
Nested and multi-threaded transactions

\[ l_3 \]

\[ l_1 \]

\[ l_2 \]

\[ l_1 : \log'_4 \]
Goal & complications

Goal

Static estimation on upper bounds of resource consumption

- memory consumption = number of transactions potentially running at in parallel × local resource consumption

- challenges
  - “concurrent” analysis (≠ safe-commits ... iFM’10, FSEN’10 [Mai Thuong Tran and Steffen, 2010, Johnsen et al., 2012])
  - implicit join-synchronization via commits (≠ “Resource bounds for components” (ICTAC’05, FMOODS’05 [Truong, 2005, Truong and Bezem, 2005] ...)
  - multithreading and nested transactions ⇒ parent-child relationship between threads relevant
Challenges

- **compositional**, syntax directed analysis
  ⇒ “interface information”
  - e.g., nesting depth (cf. “safe commit”):
    - “**single threaded**”: pre and post are enough
      \[
      n \vdash \text{commit} :: n - 1
      \]
      \[
      n_1 \vdash e_1 :: n_2 \quad n_2 \vdash e_2 :: n_3
      \]
      \[
      \overline{n_1 \vdash e_1 ; e_2 :: n_3}
      \]
  - parallel execution
Challenges

- compositional, syntax directed analysis
  ⇒ “interface information”
- e.g., nesting depth (cf. “safe commit”):
- parallel execution
  - \( P_1 \parallel P_2 \): without synchronization

\[
\because P_1 :: t_1 \quad \because P_2 :: t_2 \\
\therefore P_1 \parallel P_2 : t_1 + t_2
\]
Challenges

- compositional, syntax directed analysis
- "interface information"
- e.g., nesting depth (cf. "safe commit"): parallel execution
  - \[ \parallel \] without synchronization
  \[
  \vdash P_1 :: t_1 \quad \vdash P_2 :: t_2 \\
  \vdash P_1 \parallel P_2 : t_1 + t_2
  \]
- \[ ; \] explicit sequentialization/join
  \[
  \vdash P_1 :: t_1 \quad \vdash P_2 :: t_2 \\
  \vdash P_1; P_2 : t_1 \lor t_2
  \]
Challenges

- **compositional**, syntax directed analysis
  ⇒ “interface information”

- e.g., nesting depth (cf. “safe commit”):
- parallel execution

  **here:**

  - neither independent parallelism nor full sequentialization
  - **implicit** join synchronization via commits

  \[(\text{spawn } e_1); e_2\]
onacid; // thread 0 (main thread)
onacid;
spawn (e₁; commit; commit); // thread 1
onacid;
spawn (e₂; commit; commit; commit); // thread 2
commit;
e₃
commit;
e₄;

in the following:
onacid ⇒ [
commit ⇒ ]

\[
e₁ = [ ; [ ; [ ; \ldots ; ] ; ] ; ] = [^{3} ; \ldots ; ]^{3}
\]

\[
e₂ = [^{4} ; \ldots ; ]^{4}
\]

\[
e₃ = [^{5} ; \ldots ; ]^{5}
\]

\[
e₄ = [^{6} ; \ldots ; ]^{6}
\]
Seq. composition & Joining commit

\[
\begin{align*}
\text{[ } & - \text{ [ [ [ e_1 ] ] e_4 ]} \\
& \downarrow e_2 \downarrow e_3 \\
& \text{[ [ [ e_2 ] ] e_4 ]}
\end{align*}
\]
Seq. composition & Joining commit

\[
\begin{align*}
\text{[} & e_1 \rightarrow e_2 \rightarrow [ \quad e_3 \quad ] \rightarrow e_4 \rightarrow e_r \rightarrow e_l \text{]}
\end{align*}
\]
Seq. composition & Joining commit
Judgment & interface information

Judgment

\[ n_1 \vdash e :: n_2, h, l, \vec{t}, S \]

- current thread
  - \( n_1 \) and \( n_2 \): balance, pre- and post-condition
  - \( h, l \): maximum/minimum during execution

- not (only) current thread

compositionality

for ; : \( S \): contribution of spawned threads after execution of \( e \)

for \( || \) : \( \vec{t} \): sequence of total weights of current + spawned threads during \( e \), separated by joining commits
Judgment & interface information

**Judgment**

\[ n_1 \vdash e :: n_2, h, l, \vec{t}, S \]

- current thread
  - \( n_1 \) and \( n_2 \): balance, pre- and post-condition
  - \( h, l \): maximum/minimum *during* execution
- not (only) current thread

**compositionality**

\[ \text{for } ; : S: \text{ contribution of } \text{spawned} \text{ threads after execution of } e \]
\[ \text{for } || : \vec{t}: \text{ sequence of } \text{total} \text{ weights of current + spawned threads during } e, \text{ separated by joining commits} \]
Sample derivation: pre- and post

\[ 0 \vdash [ [ \text{spawn} (e_1) ] ] :: 2 \]

\[ 2 \vdash [ [ \text{spawn} (e_2) ] ] ; [ ] ; e_3 ] ; e_4 :: 1 \]

\[ 0 \vdash [ [ \text{spawn} (e_1) ] ] ; [ [ \text{spawn} (e_2) ] ] ; [ ] ; e_3 ] ; e_4 :: 1 \]
Sample derivation (high and low)

\[
\begin{align*}
0 \vdash & \text{[ } [ ; \text{spawn (} e_1 \text{) } ] ] :: 2, 0 \\
2 \vdash & \text{[ } [ ; \text{spawn (} e_2 \text{) } ] ] ; ] ; e_3 ] ; e_4 :: 7, 1 \\
0 \vdash & \text{[ } [ ; \text{spawn (} e_1 ; ] ] ; [ ; \text{spawn (} e_2 ; ] ] ] ; ] ; e_3 ] ; e_4 :: 7, 0
\end{align*}
\]
Sample derivation (par. contribution and synchronization)

\[
0 \vdash [ [ ; \text{spawn } (e_1) ] ] :: [7], \{(2, 3)\} \quad 2 \vdash [ [ ; \text{spawn } (e_2) ] ] ; [ ] ; e_3 ] ; e_4 :: [10, 8], \{(1, 0)\}
\]

\[
0 \vdash [ [ ; \text{spawn } (e_1) ] ] ; [ [ ; \text{spawn } (e_2) ] ] ; [ ] ; e_3 ] ; e_4 :: t, \{(1, 0), (1, 0)\}
\]

\[
t = 7 \lor (10 + |\{(2, 3)\}|) \lor (8 + |\{(1, 0)\}|)
\]
Sample derivation: different split

\[
0 \vdash [^2; \text{spawn } e_1; [ ; (\text{spawn } e_2); ] :: [15], \{(2, 3), (0, 2)\}] \\
2 \vdash e_3; ] ; e_4 :: [7, 7], \{
\]

\[
0 \vdash [^2; \text{spawn } e_1; [ ; (\text{spawn } e_2); ] ; e_3 ] ; e_4 :: 1, 7, 0, t, \{(1, 0), (1, 0)\}
\]
Sequential composition

\[
\begin{align*}
n_1 & \vdash e_1 :: n_2, h_1, l_1, \bar{s}, S_1 \\
n_2 & \vdash e_2 :: n_3, h_2, l_2, \bar{t}, S_2 \\
h & = h_1 \lor h_2 \\
l & = l_1 \land l_2 \\
p & = n_2 - l_1 \\
S & = S_1 \downarrow_{l_2} \cup S_2 \\
\bar{u} & = \bar{s} \oplus_p (S_1 \ominus n_2 \bar{t}) \\
n_1 \vdash \text{let } x : T = e_1 \text{ in } e_2 :: n_3, h, l, \bar{u}, S
\end{align*}
\]
Sequential composition

\[ n_1 \vdash e_1 :: n_2, h_1, l_1, \vec{s}, S_1 \quad n_2 \vdash e_2 :: n_3, h_2, l_2, \vec{t}, S_2 \]
\[ h = h_1 \lor h_1 \quad l = l_1 \land l_2 \]
\[ \vec{s} = s_1, \ldots, s_k \quad \vec{t} = t_1, \ldots, t_m \quad k, m \geq 1 \quad p = n_2 - l_1 \]
\[ t'_1 = t_1 + |S_1| \quad t'_2 = t_2 + |S_1 \downarrow_{n_2-1}| \quad t'_3 = t_3 + |S_1 \downarrow_{n_2-2}| \quad \ldots \]
\[ S = S_1 \downarrow_{l_2} \cup S_2 \]
\[ \vec{u} = s_1, \ldots, s_{k-1}, s_k \lor t'_1 \lor \ldots \lor t'_p, t'_{p+1}, \ldots, t'_m \]
\[ n_1 \vdash e_1; e_2 :: n_3, h, l, \vec{u}, S \]
Parallel composition

- similarly complex
- merging trees / forests using join-commits-labels
- using tree representation of future joining commit behavior
Parallel composition

- similarly complex ("hidden" in def. of $\otimes$)
- merging trees / forests using join-commits-labels
- using tree representation of future joining commit behavior $t_1$ and $t_2$

\[
\Gamma_1 \vdash P_1 : t_1 \quad \Gamma_2 \vdash P_2 : t_2 \\
\hline
\Gamma_1, \Gamma_2 \vdash P_1 \parallel P_2 : t_1 \otimes t_2
\]

$T\text{-PAR}$
Results and future work

**Soundness**

Soundness of the analysis: “subject reduction”

- higher-order functions
- type inference
- machine checked proof of SR (Coq/OTT)
- different synchronization model


