1 Semantics of concurrent revisions

In concurrent revisions, each thread gets its own isolated thread of control. All updates to "shared" state occur at joins.

Contributions

1. Semantics
2. Proof of determinancy
3. Formalizes connection to snapshot isolation

```
x=0
y=0
r = rfork { if y=0 then x++ }
if x=0 then y++;
rjoin r;
```

Assuming versioned types for x and y, output is $x == 1$ and $y == 1$.

2 Encoding transactions in concurrent revisions

Grammar

$$
\begin{aligned}
v &::= \quad c \mid x \mid l \mid r \mid \lambda x.e \\
e &::= \quad v \mid ee \mid (e?\, e : e) \mid \texttt{ref}\ \ e \mid \\
&\quad\ !e \mid e := e \mid \texttt{rfork}\ \ e \mid \texttt{rjoin}\ \ e \\
\varepsilon &::= \quad \square \mid \varepsilon\ e \mid v\ \varepsilon \mid (\varepsilon?\, e : E) \mid \texttt{ref}\ \ e \\
&\quad\ !\varepsilon \mid \varepsilon := e \mid l := \varepsilon \mid \texttt{rjoin}\ \ \varepsilon
\end{aligned}
$$

$s \in Rid \rightharpoonup \texttt{Store} \times \texttt{Store} \times \texttt{Expr}$

Selected rules

Definition.

$$s \rightarrow_r s'$$

Definition (Apply)**.**

$$s\ (r \mapsto \langle \sigma, \tau, \varepsilon[(\lambda x.e)v]\rangle]) \rightarrow_r s\ [r \mapsto \langle \sigma, \tau, \varepsilon[e[v/x]]\rangle]$$

Definition (Deref)**.**

$$s\ (r \mapsto \langle \sigma, \tau, \varepsilon[!\ell]\rangle]) \rightarrow_r s\ [r \mapsto \langle \sigma, \tau, \varepsilon[(\sigma :: \tau)(\ell)]\rangle]$$

Definition (Fork)**.**

$$s\ (r \mapsto \langle \sigma, \tau, \varepsilon[\texttt{rfork}\ \ e]\rangle]) \rightarrow_r s\ [r \mapsto \langle \sigma, \tau, \varepsilon[r']\rangle \quad r' \mapsto \langle \sigma :: \tau, \cdot, e\rangle]$$

Definition (Join)**.**

$$\begin{array}{rc} s\ [r \mapsto \langle \sigma, \tau, \varepsilon[\texttt{rjoin}\ \ r']\rangle \quad r' \mapsto \langle \sigma', \tau', v\rangle] \rightarrow_r & s[r \mapsto \langle \sigma, \tau :: \tau', \varepsilon[v]\rangle, r' \mapsto \bot] \\ s\ [r \mapsto \langle \sigma, \tau, \varepsilon[\texttt{rjoin}\ \ r']\rangle \quad r' \mapsto \langle \sigma', \tau', \bot\rangle] \rightarrow_r & \texttt{error} \end{array}$$

Notation $e \downarrow s$ Evaluating e results in the store s

Definition. Equivalence
$s \approx s' \iff \exists \alpha, \beta . s = \alpha(\beta(s'))$
Where α and β rewrite thread ids and location ids.

Theorem If $e \downarrow s$ and $e \downarrow s'$ then $s \approx s'$

Lemma $\rightarrow_r$ preserves $\approx$

Lemma If s_1, s_1' are reachable and $s_1 \approx s_1'$ and

$$\begin{array}{lcl} s_1 & \rightarrow_r & s_2 \\ s_1' & \rightarrow_r' & s_2' \end{array}$$

then $\exists s_3, s_3'$

$$\begin{array}{lcl} s_2 & \rightarrow_r' & s_3 \\ s_2' & \rightarrow_r & s_3' \end{array}$$

and $s_3 \approx s_3'$