Towards a Grainless Semantics for Shared Variable Concurrency

John C. Reynolds

Carnegie Mellon University
Queen Mary, University of London
Imperial College

December 14, 2010

This is a preliminary draft describing work in progress

The Problem

What is the meaning of

$$x := x \times x \parallel x := x + 1\,?$$

Are the assignment commands atomic, so that it is either

$$x := x \times x\,;\, x := x + 1 \quad \text{or} \quad x := x + 1\,;\, x := x \times x\,?$$

or are evaluation and store operations atomic:

$$(t_1 := x \times x\,;\, x := t_1) \parallel (t_2 := x + 1\,;\, x := t_2)\,?$$

or is each lookup and store atomic:

$$(t_1 := x\,;\, t_2 := x\,;\, x := t_1 \times t_2) \parallel (t_3 := x\,;\, x := t_3 + 1)\,?$$

or is the granularity even finer:

$$
\begin{aligned}
&(t_1{}^{low} := x^{low}\,;\ t_1{}^{up} := x^{up}\,;\ t_2{}^{low} := x^{low}\,;\ t_2{}^{up} := x^{up}\,; \\
&\qquad x^{low} := (t_1 \times t_2)^{low}\,;\ x^{up} := (t_1 \times t_2)^{up}) \parallel \\
&(t_3{}^{low} := x^{low}\,;\ t_3{}^{up} := x^{up}\,; \\
&\qquad x^{low} := (t_3 + 1)^{low}\,;\ x^{up} := (t_3 + 1)^{up})\ ?
\end{aligned}
$$

An Early Answer

In the early 70's, Hoare and Brinch-Hansen claimed that constructions such as

$$\mathsf{x} := \mathsf{x} \times \mathsf{x} \parallel \mathsf{x} := \mathsf{x} + 1$$

should be syntactically illegal.

Instead, when the same variable appears on both sides of $\parallel$, the programmer should be required to indicate the appropriate mutual exclusion explicitly by means of critical regions.

For example,

$$\textbf{with } \mathsf{lock} \textbf{ do } \mathsf{x} := \mathsf{x} \times \mathsf{x} \parallel \textbf{with } \mathsf{lock} \textbf{ do } \mathsf{x} := \mathsf{x} + 1$$

or

$$\begin{aligned}&(\textbf{with } \mathsf{lock} \textbf{ do } \mathsf{t}_1 := \mathsf{x}\,;\, \textbf{with } \mathsf{lock} \textbf{ do } \mathsf{x} := \mathsf{t}_1 \times \mathsf{t}_1) \parallel \\ &\quad (\textbf{with } \mathsf{lock} \textbf{ do } \mathsf{t}_2 := \mathsf{x}\,;\, \textbf{with } \mathsf{lock} \textbf{ do } \mathsf{x} := \mathsf{t}_2 + 1).\end{aligned}$$

The Harder Problem

What about lookup and store via pointers,

$$[\mathsf{x}] := [\mathsf{x}] \times [\mathsf{x}] \parallel [\mathsf{y}] := [\mathsf{y}] + 1,$$

where aliasing cannot be decided by a compiler.

Our Answer

When the addresses x and y are equal, the meaning of the above program is simply "**wrong**".

No further information makes sense at any level of abstraction above the machine-language implementation.

Four Principles for Grainless Concurrency

- All operations except locking and unlocking have duration, and can overlap one another during execution.
- If two overlapping operations lookup or set the same location, the meaning of program execution is $\mathbf{wrong}$.
- If, from a given starting state, execution of a program can give $\mathbf{wrong}$, then no other possibilities need be considered.
- The meaning of a command is a *trace tree*, i.e., a tree whose paths are traces.

History

- Trace semantics: D. M. R. Park 1980, S. D. Brookes 1996
- Concurrent Separation Logic: P. W. O'Hearn 2004
- Soundness of Concurrent Separation Logic: Brookes 2004
- Small-Step Grainless Semantics: Reynolds 2004
- Large-Step Grainless Semantics: Brookes 2005
- Grainless Semantics without Synchronization: Reynolds 2007

Examples

$$\mathsf{y} := \mathsf{x} - \mathsf{x} \;\not\simeq\; \mathsf{y} := 0$$

$$\mathsf{x} := \mathsf{x} + 1\,;\, \mathsf{x} := \mathsf{x} + 2 \;\simeq\; \mathsf{x} := \mathsf{x} + 3$$

$$\mathsf{x} := \mathsf{x} + 1\,;\, \mathsf{y} := \mathsf{y} + 2 \;\simeq\; \mathsf{y} := \mathsf{y} + 2\,;\, \mathsf{x} := \mathsf{x} + 1$$

$$[\mathsf{x}] := [\mathsf{x}] + 1\,;\, [\mathsf{y}] := [\mathsf{y}] + 2 \;\simeq\; [\mathsf{y}] := [\mathsf{y}] + 2\,;\, [\mathsf{x}] := [\mathsf{x}] + 1$$

$$\mathsf{x} := 0 \;\mathbf{or}\; \mathsf{y} := 0 \;\simeq\; (\mathsf{x} := 0\,;\, \mathsf{y} := \mathsf{y}) \;\mathbf{or}\; (\mathsf{y} := 0\,;\, \mathsf{x} := \mathsf{x})$$

$$\mathsf{x} := \mathsf{x} \times \mathsf{x}\,;\, \mathbf{with}\; \mathsf{r}\; \mathbf{do}\; (\mathsf{x} := \mathsf{x} + 1\,;\, \mathsf{y} := 0) \;\simeq\; \mathsf{x} := \mathsf{x} \times \mathsf{x} + 1\,;\, \mathbf{with}\; \mathsf{r}\; \mathbf{do}\; \mathsf{y} := 0$$

An Overview

Programming Language

↓ $[\![-]\!]$ (compositional)

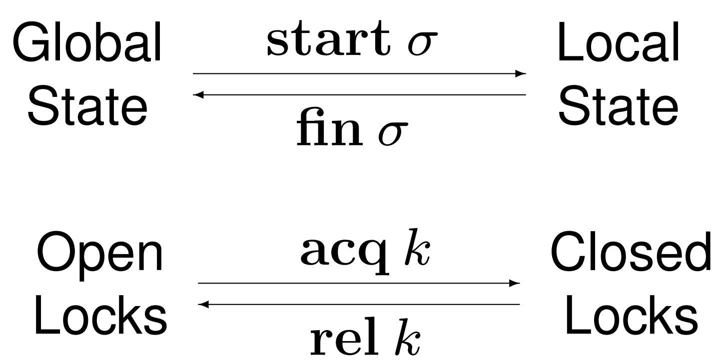
Trace Trees

↓ Ex (noncompositional)

Nondeterministic State-Transition Functions

An Overview (continued)

- The paths of *trace trees* are *traces*.
- A trace is a sequence of *actions*.
- $\mathbf{start}\,\sigma$. $\mathbf{fin}\,\sigma$. $\mathbf{acq}\,k$, $\mathbf{rel}\,k$ (and others) are actions.



A Simple Example

The meaning $[\![\mathsf{x} := \mathsf{x} \times \mathsf{x}]\!]$ is a tree whose traces are

$$\mathbf{start}[\,\mathsf{x}{:}\, n\,]\;\mathbf{fin}[\,\mathsf{x}{:}\, n \times n\,],$$

for all integers n, which includes, for example, the trace

$$\mathbf{start}[\,\mathsf{x}{:}\, 3\,]\;\mathbf{fin}[\,\mathsf{x}{:}\, 9\,].$$

If $\mathsf{x}{:=}\mathsf{x}{\times}\mathsf{x}$ is not running concurrently, its traces can be executed sequentially. Starting in the global state $[\,\mathsf{x}{:}\, 3 \mid \mathsf{t}{:}\, 22\,]$,

$$\begin{array}{ll} [\,\mathsf{x}{:}\, 3 \mid \mathsf{t}{:}\, 22\,] & \\ \quad\downarrow & \mathbf{start}[\,\mathsf{x}{:}\, 3\,] \\ [\,\mathsf{t}{:}\, 22\,] & \\ \quad\downarrow & \mathbf{fin}[\,\mathsf{x}{:}\, 9\,] \\ [\,\mathsf{x}{:}\, 9 \mid \mathsf{t}{:}\, 22\,] & \end{array}$$

On the other hand, starting in the global state $[\,\mathsf{x}\colon 4 \mid \mathsf{t}\colon 22\,]$, the trace

$$\mathbf{start}[\,\mathsf{x}\colon 3\,]\,\mathbf{fin}[\,\mathsf{x}\colon 9\,]$$

has no execution. But an execution is provided by another trace in the same tree, $\mathbf{start}[\,\mathsf{x}\colon 4\,]\,\mathbf{fin}[\,\mathsf{x}\colon 16\,]$ in $[\![\mathsf{x} := \mathsf{x} \times \mathsf{x}]\!]$:

$$\begin{array}{ll} [\,\mathsf{x}\colon 4 \mid \mathsf{t}\colon 22\,] & \\ \quad\downarrow & \qquad \mathbf{start}[\,\mathsf{x}\colon 4\,] \\ [\,\mathsf{t}\colon 22\,] & \\ \quad\downarrow & \qquad \mathbf{fin}[\,\mathsf{x}\colon 16\,] \\ [\,\mathsf{x}\colon 16 \mid \mathsf{t}\colon 22\,]. & \end{array}$$

A third possibility arises when x does not occur in the domain of the $\mathbf{start}$ state:

$$\begin{array}{ll} [\,\mathsf{t}\colon 22\,] & \\ \quad\downarrow & \mathbf{start}[\,\mathsf{x}\colon 3\,] \\ \mathbf{wrong}. & \end{array}$$

An Example of Pointers and Concurrency

The meaning $[\![[\mathsf{x}] := [\mathsf{x}] \times [\mathsf{x}]]\!]$ is the tree with traces

$$\mathbf{start}[\,\mathsf{x}\colon n_1 \mid n_1\colon n_2\,]\,\mathbf{fin}[\,\mathsf{x}\colon n_1 \mid n_1\colon n_2 \times n_2\,],$$

for all integers n_1 and n_2.

Similarly, the meaning $[\![[\mathsf{y}] := [\mathsf{y}] + 1]\!]$ has the traces

$$\mathbf{start}[\,\mathsf{y}\colon n_3 \mid n_3\colon n_4\,]\,\mathbf{fin}[\,\mathsf{y}\colon n_3 \mid n_3\colon n_4 + 1\,],$$

for all integers n_3 and n_4.

The meaning $[\![[\mathsf{x}] := [\mathsf{x}] \times [\mathsf{x}] \parallel [\mathsf{y}] := [\mathsf{y}] + 1]\!]$ is a tree containing the interleavings of the traces of each subcommand:

$$\begin{aligned}
&\mathbf{start}[\,\mathsf{x}\colon n_1 \mid n_1\colon n_2\,]\, \mathbf{fin}[\,\mathsf{x}\colon n_1 \mid n_1\colon n_2 \times n_2\,] \\
&\qquad \mathbf{start}[\,\mathsf{y}\colon n_3 \mid n_3\colon n_4\,]\, \mathbf{fin}[\,\mathsf{y}\colon n_3 \mid n_3\colon n_4 + 1\,], \\
&\mathbf{start}[\,\mathsf{x}\colon n_1 \mid n_1\colon n_2\,]\, \mathbf{start}[\,\mathsf{y}\colon n_3 \mid n_3\colon n_4\,] \\
&\qquad \mathbf{fin}[\,\mathsf{x}\colon n_1 \mid n_1\colon n_2 \times n_2\,]\, \mathbf{fin}[\,\mathsf{y}\colon n_3 \mid n_3\colon n_4 + 1\,], \\
&\mathbf{start}[\,\mathsf{x}\colon n_1 \mid n_1\colon n_2\,]\, \mathbf{start}[\,\mathsf{y}\colon n_3 \mid n_3\colon n_4\,] \\
&\qquad \mathbf{fin}[\,\mathsf{y}\colon n_3 \mid n_3\colon n_4 + 1\,]\, \mathbf{fin}[\,\mathsf{x}\colon n_1 \mid n_1\colon n_2 \times n_2\,], \\
&\qquad \vdots
\end{aligned}$$

which includes traces where the two assignment operations overlap.

Consider the overlapping trace

$$\mathbf{start}[\,\mathsf{x}\colon n_1 \mid n_1\colon n_2\,]\,\mathbf{start}[\,\mathsf{y}\colon n_3 \mid n_3\colon n_4\,]$$
$$\mathbf{fin}[\,\mathsf{y}\colon n_3 \mid n_3\colon n_4+1\,]\,\mathbf{fin}[\,\mathsf{x}\colon n_1 \mid n_1\colon n_2 \times n_2\,].$$

When $n_1 \neq n_3$, there is no interference, and we have

$$\begin{array}{ll}
[\,\mathsf{x}\colon n_1 \mid n_1\colon n_2 \mid \mathsf{y}\colon n_3 \mid n_3\colon n_4\,] & \\
\quad\downarrow & \mathbf{start}[\,\mathsf{x}\colon n_1 \mid n_1\colon n_2\,] \\
[\,\mathsf{y}\colon n_3 \mid n_3\colon n_4\,] & \\
\quad\downarrow & \mathbf{start}[\,\mathsf{y}\colon n_3 \mid n_3\colon n_4\,] \\
[\,] & \\
\quad\downarrow & \mathbf{fin}[\,\mathsf{y}\colon n_3 \mid n_3\colon n_4+1\,] \\
[\,\mathsf{y}\colon n_3 \mid n_3\colon n_4+1\,] & \\
\quad\downarrow & \mathbf{fin}[\,\mathsf{x}\colon n_1 \mid n_1\colon n_2 \times n_2\,] \\
[\,\mathsf{y}\colon n_3 \mid n_3\colon n_4+1 \mid \mathsf{x}\colon n_1 \mid n_1\colon n_2 \times n_2\,]. &
\end{array}$$

On the other hand, when $n_1 = n_3$ (which implies $n_2 = n_4$), the two processes interfere, as shown by the execution

$$
\begin{array}{ll}
[\,\mathsf{x}\colon n_1 \mid n_1\colon n_2 \mid \mathsf{y}\colon n_1\,] & \\
\quad\downarrow & \mathbf{start}[\,\mathsf{x}\colon n_1 \mid n_1\colon n_2\,] \\
[\,\mathsf{y}\colon n_1\,] & \\
\quad\downarrow & \mathbf{start}[\,\mathsf{y}\colon n_1 \mid n_1\colon n_2\,] \\
\mathsf{wrong}. &
\end{array}
$$

Locks

To deal with critical regions, we move from global states to configurations $\langle K, \sigma \rangle$, where

K is a set of closed (i.e., acquired) locks,
σ is a global state.

We also add the actions

$\mathbf{acq}\, k$ to acquire the lock k,
$\mathbf{rel}\, k$ to release the lock k.

An Example of a Critical Region

The meaning $[\![\mathbf{with}\ \mathsf{k}\ \mathbf{do}\ \mathsf{x} := \mathsf{x} \times \mathsf{x}]\!]$ contains the trace

$$\mathbf{acq}\,\mathsf{k}\ \mathbf{start}[\,\mathsf{x}\colon 3\,]\ \mathbf{fin}[\,\mathsf{x}\colon 9\,]\ \mathbf{rel}\,\mathsf{k}.$$

Suppose the lock k2 was already closed. Then

$$\begin{array}{rl}
\{\mathsf{k2}\}, [\,\mathsf{x}\colon 3 \mid \mathsf{t}\colon 22\,] & \\
\downarrow & \mathbf{acq}\,\mathsf{k} \\
\{\mathsf{k2}, \mathsf{k}\}, [\,\mathsf{x}\colon 3 \mid \mathsf{t}\colon 22\,] & \\
\downarrow & \mathbf{start}[\,\mathsf{x}\colon 3\,] \\
\{\mathsf{k2}, \mathsf{k}\}, [\,\mathsf{t}\colon 22\,] & \\
\downarrow & \mathbf{fin}[\,\mathsf{x}\colon 9\,] \\
\{\mathsf{k2}, \mathsf{k}\}, [\,\mathsf{x}\colon 9 \mid \mathsf{t}\colon 22\,] & \\
\downarrow & \mathbf{rel}\,\mathsf{k} \\
\{\mathsf{k2}\}, [\,\mathsf{x}\colon 9 \mid \mathsf{t}\colon 22\,] &
\end{array}$$

The Programming Language

We begin with the simple imperative language:

$$\begin{aligned}
\langle \mathsf{exp} \rangle &::= \langle \mathsf{var} \rangle \mid \langle \mathsf{constant} \rangle \mid \langle \mathsf{exp} \rangle + \langle \mathsf{exp} \rangle \mid \cdots \\
\langle \mathsf{bexp} \rangle &::= \langle \mathsf{exp} \rangle = \langle \mathsf{exp} \rangle \mid \cdots \mid \langle \mathsf{bexp} \rangle \wedge \langle \mathsf{bexp} \rangle \mid \cdots \\
\langle \mathsf{comm} \rangle &::= \langle \mathsf{var} \rangle := \langle \mathsf{exp} \rangle \mid \mathbf{skip} \mid \langle \mathsf{comm} \rangle \, ; \langle \mathsf{comm} \rangle \\
&\quad\mid \mathbf{if}\ \langle \mathsf{bexp} \rangle\ \mathbf{then}\ \langle \mathsf{comm} \rangle\ \mathbf{else}\ \langle \mathsf{comm} \rangle \\
&\quad\mid \mathbf{while}\ \langle \mathsf{bexp} \rangle\ \mathbf{do}\ \langle \mathsf{comm} \rangle
\end{aligned}$$

and add lookup and mutation operations:

$$\langle \mathsf{exp} \rangle ::= [\langle \mathsf{exp} \rangle]$$

$$\langle \mathsf{comm} \rangle ::= [\langle \mathsf{exp} \rangle] := \langle \mathsf{exp} \rangle$$

concurrent composition:

$$\langle \mathsf{comm} \rangle ::= \langle \mathsf{comm} \rangle \parallel \langle \mathsf{comm} \rangle$$

and critical regions:

$$\begin{aligned} \langle \mathsf{comm} \rangle ::= {} & \mathbf{with}\ \langle \mathsf{lock} \rangle\ \mathbf{do}\ \langle \mathsf{comm} \rangle \\ & \mid \mathbf{with}\ \langle \mathsf{lock} \rangle\ \mathbf{when}\ \langle \mathsf{bexp} \rangle\ \mathbf{do}\ \langle \mathsf{comm} \rangle \end{aligned}$$

What is Missing

- Unbounded Nondeterminism
- Allocation and disposal.
- Passivity (i.e., allowing read-only interference).

States

$$\begin{aligned} \mathsf{Addresses} &\subseteq \mathcal{Z} \\ \mathsf{Locations} &= \langle \mathsf{var} \rangle \uplus \mathsf{Addresses} \\ \mathsf{States} &= \bigcup \{\, \delta \to \mathcal{Z} \mid \delta \overset{\mathsf{fin}}{\subseteq} \mathsf{Locations} \,\}. \end{aligned}$$

We write:

- $\sigma \smile \sigma'$ when states σ and σ' are *compatible*, i.e., when $\sigma \cup \sigma'$ is a function, or equivalently, when σ and σ' agree on the intersection of their domains.
- $\delta \perp \delta'$ when the sets δ and δ' are disjoint.
- $\sigma \perp \sigma'$ when $\operatorname{dom}\sigma \perp \operatorname{dom}\sigma'$.
- $[\,\sigma \mid \ell\colon n\,]$ for the state such that

$$\operatorname{dom}[\,\sigma \mid \ell\colon n\,] = \operatorname{dom}\sigma \cup \{\ell\}$$
$$[\,\sigma \mid \ell\colon n\,](\ell) = n$$
$$[\,\sigma \mid \ell\colon n\,](\ell') = \sigma(\ell') \text{ when } \ell \neq \ell'.$$

Semantics of Expressions

$$[\![\langle\mathsf{exp}\rangle]\!] \in \mathsf{States} \rightharpoonup \mathcal{Z} \qquad [\![\langle\mathsf{bexp}\rangle]\!] \in \mathsf{States} \rightharpoonup \mathcal{B}$$

$$[\![0]\!] = \{\langle[\,], 0\rangle\}$$

$$[\![\mathsf{x} - \mathsf{x}]\!] = \{\, \langle[\,\mathsf{x}\colon m\,], 0\rangle \mid m \in \mathcal{Z}\,\}$$

$$[\![\mathsf{x} + [\mathsf{y}]]\!] = \{\, \langle[\,\mathsf{x}\colon m \mid \mathsf{y}\colon n \mid n\colon n'\,], m + n'\rangle \mid m, n, m' \in \mathcal{Z}\,\}.$$

The value of e in a state σ is $[\![e]\!]\sigma_f$, where $\sigma_f \in \mathsf{dom}[\![e]\!]$ is a subset of σ. If there is no such σ_f, then the evaluation aborts.

To avoid nondeterminism, we require

$$\forall \sigma_f,\ \sigma_f' \in \mathsf{dom}[\![e]\!].\ \sigma_f \neq \sigma_f' \Rightarrow \sigma_f \not\smile \sigma_f'.$$

The $\sigma_f \in \mathsf{dom}[\![e]\!]$ are the *footprints* of e.

A Grammar for Trace Trees

$$\langle\text{tree}\rangle ::= \mathbf{halt}\ \langle\text{label}\rangle \mid \mathbf{dlock}$$

$$\left|\ \mathbf{start} \begin{cases} \langle\text{state}\rangle\ \langle\text{tree}\rangle \\ \langle\text{state}\rangle\ \langle\text{tree}\rangle \\ \quad\vdots \end{cases}\right| \mathbf{fin}\ \langle\text{state}\rangle\ \langle\text{tree}\rangle$$

$$\mid \mathbf{acq}\ \langle\text{lock}\rangle\ \langle\text{tree}\rangle \mid \mathbf{rel}\ \langle\text{lock}\rangle\ \langle\text{tree}\rangle$$

$$\left|\ \mathbf{or} \begin{cases} \langle\text{tree}\rangle \\ \langle\text{tree}\rangle \\ \quad\vdots \end{cases}\right.$$

Trace trees may have infinite paths, and start nodes may have infinitely many immediate subnodes. "**or**" nodes may have infinitely many subnodes iff unbounded nondeterminism is allowed.

The $\mathbf{start}$ Node

The subtrees of a $\mathbf{start}$ node form a (usually) infinite set. In general, we write

$$\mathbf{start}\begin{cases}\mathbf{for}\ i \in S_1 \colon \sigma_{1i}\,\tau_{1i} \\ \quad\vdots \\ \mathbf{for}\ i \in S_n \colon \sigma_{ni}\,\tau_{ni}\end{cases} \text{ to abbreviate } \mathbf{start}\begin{cases}\{\, \sigma_{1i}\,\tau_{1i} \mid i \in S_1 \,\} \\ \quad \uplus \cdots \uplus \\ \{\, \sigma_{ni}\,\tau_{ni} \mid i \in S_n \,\}.\end{cases}$$

We require that, if σ and σ' occur at the beginning of distinct subtrees of a $\mathbf{start}$ node, then $\sigma \not\sim \sigma'$. This insures that $\mathbf{start}$ nodes are determinate. We also write

$$\mathbf{abort} \quad \text{to abbreviate} \quad \mathbf{start}\big\{\, \{\}.$$

The $\mathbf{or}$ Node

$\mathbf{or}$ nodes describe nondeterminism. We write

$$\mathbf{or}\left\{\begin{array}{l}\mathbf{for}\ i \in S_1 \colon \tau_{1i} \\ \qquad \vdots \\ \mathbf{for}\ i \in S_n \colon \tau_{ni}\end{array}\right. \quad \text{to abbreviate} \quad \mathbf{or}\left\{\begin{array}{c}\{\,\tau_{1i} \mid i \in S_1\,\} \\ \cup \cdots \cup \\ \{\,\tau_{ni} \mid i \in S_n\,\}.\end{array}\right.$$

These nodes are quite different from $\mathbf{start}$ nodes. Adding subnodes to a $\mathbf{start}$ node can only reduce the possibilities for aborting. But adding subnodes to an $\mathbf{or}$ node can only increase the possibilities for aborting.

Note that an $\mathbf{or}$ node leaves no action in the traces that pass through it.

Examples

$$[\![\mathsf{x} := \mathsf{x}+1]\!] = \mathbf{start}\left\{\ \mathbf{for}\ n \in \mathcal{Z}\colon\ [\,\mathsf{x}\colon n\,]\ \mathbf{fin}\ [\,\mathsf{x}\colon n+1\,]\ \mathbf{halt}\ 0\right.$$

$$[\![\mathbf{if}\ \mathsf{y}=0\ \mathbf{then}\ \mathsf{x} := 1\ \mathbf{else}\ \mathbf{skip}]\!] = \mathbf{start}\left\{\begin{array}{l}[\,\mathsf{y}\colon 0\,]\ \mathbf{fin}[\,\mathsf{y}\colon 0\,]\ \mathbf{start}\left\{\ \mathbf{for}\ n \in \mathcal{Z}\colon [\,\mathsf{x}\colon n\,]\ \mathbf{fin}\ [\,\mathsf{x}\colon 1\,]\right. \\ \qquad \mathbf{halt}\ 0 \\ \\ \mathbf{for}\ m \in \mathcal{Z}-\{0\}\colon [\,\mathsf{y}\colon m\,]\ \mathbf{fin}\ [\,\mathsf{y}\colon m\,]\mathbf{halt}\ 0\end{array}\right.$$

or better:

$$[\![\mathbf{if}\ \mathsf{y}=0\ \mathbf{then}\ \mathsf{x} := 1\ \mathbf{else}\ \mathbf{skip}]\!] = \mathbf{start}\left\{\begin{array}{l}\mathbf{for}\ n \in \mathcal{Z}\colon [\,\mathsf{y}\colon 0 \mid \mathsf{x}\colon n\,]\ \mathbf{fin}\ [\,\mathsf{y}\colon 0 \mid \mathsf{x}\colon 1\,]\ \mathbf{halt}\ 0 \\ \\ \mathbf{for}\ m \in \mathcal{Z}-\{0\}\colon [\,\mathsf{y}\colon m\,]\ \mathbf{fin}\ [\,\mathsf{y}\colon m\,]\mathbf{halt}\ 0\end{array}\right.$$

Useless Traces

Traces such as:

$$\cdots\ \mathbf{fin}\,[\,\mathsf{x}\colon 0\,]\,\mathbf{fin}\,[\,\mathsf{x}\colon 1\,]\ \cdots \qquad\qquad \cdots\ \mathbf{rel}\,k\,\mathbf{rel}\,k\ \cdots$$

can never occur as the meaning of programs, because $\mathbf{start}/\mathbf{fin}$ and $\mathbf{acq}/\mathbf{rel}$ actions are balanced. Also, traces such as:

$\cdots\ \mathbf{acq}\,k\,\mathbf{acq}\,k\ \cdots$ can be replaced by $\cdots\ \mathbf{acq}\,k\,\mathbf{dlock}$

$\cdots\ \mathbf{start}\,[\,\mathsf{x}\colon 0\,]\,\mathbf{start}\,[\,\mathsf{x}\colon 1\,]\ \cdots$ can be replaced by
$\cdots\ \mathbf{start}\,[\,\mathsf{x}\colon 0\,]\,\mathbf{abort}$

To eliminate these traces, we will describe well-formed trace trees by type inference rules. To do so (in this talk), we will limit ourselves to trace trees with finite paths (by ignoring $\mathbf{while}$ commands and conditional critical regions).

Type Judgements for Trace Trees

$$\mathsf{TreeTypes} = \mathcal{P}_{\mathsf{fin}}(\mathsf{Locks}) \times \mathcal{P}_{\mathsf{fin}}(\mathsf{Locs}).$$

If a trace tree has type $\langle K, L \rangle$, then its traces can be executed starting in any configuration $\langle K', \sigma_g \rangle$ such that $K' = K$ and dom $\sigma_g \perp L$.

$$\mathsf{Contexts} = \mathsf{Labels} \rightharpoonup \mathsf{TreeTypes}.$$

If ϕ is the context $n_1 \colon \langle K_1, L_1 \rangle, \ldots, n_k \colon \langle K_k, L_k \rangle$, then

$$\phi \vdash \tau : \langle K, L \rangle$$

is a judgement that the tree τ has type $\langle K, L \rangle$ and the $\mathbf{halt}$ commands within τ have labels in $\{n_1, \ldots, n_k\}$ with the types specified by ϕ.

Formation Rules for Trace Trees

$$\frac{\phi(n) = \langle K, L\rangle}{\phi \vdash \mathbf{halt}\, n : \langle K, L\rangle} \qquad \frac{}{\phi \vdash \mathbf{dlock} : \langle K, L\rangle}$$

$$\frac{\begin{array}{c}\forall i \in S.\ L \perp \mathsf{dom}\, \sigma_i \\ \forall i, j \in S.\ i \neq j \Rightarrow \sigma_i \not\smile \sigma_j \\ \forall i \in S.\ \phi \vdash \tau_i : \langle K, L \cup \mathsf{dom}\, \sigma_i\rangle\end{array}}{\phi \vdash \left(\mathbf{start} \left\{ \mathbf{for}\ i \in S\colon \sigma_i\, \tau_i\right) : \langle K, L\rangle}$$

$$\frac{\begin{array}{c}\mathsf{dom}\, \sigma \subseteq L \\ \phi \vdash \tau : \langle K, L - \mathsf{dom}\, \sigma\rangle\end{array}}{\phi \vdash \mathbf{fin}\, \sigma\, \tau : \langle K, L\rangle}$$

Formation Rules for Trace Trees (continued)

$$\frac{k \notin K \qquad \phi \vdash \tau\langle K \cup \{k\}, L\rangle}{\phi \vdash \mathbf{acq}\, k\, \tau : \langle K, L\rangle}$$

$$\frac{k \in K \qquad \phi \vdash \tau\langle K - \{k\}, L\rangle}{\phi \vdash \mathbf{rel}\, k\, \tau : \langle K, L\rangle}$$

$$\frac{\forall i \in S.\ \phi \vdash \tau_i : \langle K, L\rangle}{\phi \vdash \left(\mathbf{or}\left\{\ \mathbf{for}\ i \in S\colon \tau_i\right) : \langle K, L\rangle}$$

Execution of Trace Trees

When $\phi \vdash \tau : \langle K, L \rangle$ and $\mathsf{dom}\, \sigma_g \perp L$,

$$\mathsf{Ex}\, \tau\, \sigma_g \subseteq \{\mathbf{wrong}, \mathbf{dlock}\} \cup \sum_{n \in \mathsf{dom}\, \phi} \{\, \sigma'_g \mid \mathsf{dom}\, \sigma'_g \perp L' \text{ where } \phi(n) = \langle K', L' \rangle \,\}.$$

$$\mathsf{Ex}\, (\mathbf{halt}\; n)\, \sigma_g = \{\langle n, \sigma_g \rangle\} \qquad \mathsf{Ex}\, (\mathbf{dlock})\, \sigma_g = \{\mathbf{dlock}\}$$

$$\mathsf{Ex}\, (\mathbf{start} \left\{ \, \mathbf{for}\; i \in S \colon \sigma_i\, \tau_i \right)\, \sigma_g = \begin{cases} \mathsf{Ex}\, \tau_i\, (\sigma_g - \sigma_i) & \text{for the unique } i \in S \text{ such that } \sigma_i \subseteq \sigma_g \\ \{\mathbf{wrong}\} & \text{if no } i \text{ satisfies } \sigma_i \subseteq \sigma_g \end{cases}$$

$$\mathsf{Ex}\, (\mathbf{fin}\; \sigma\, \tau)\, \sigma_g = \mathsf{Ex}\; \tau\, (\sigma_g \cup \sigma)$$

$$\mathsf{Ex}\, (\mathbf{acq}\; k\, \tau)\, \sigma_g = \mathsf{Ex}\; \tau\, \sigma_g \qquad \mathsf{Ex}\, (\mathbf{rel}\; k\, \tau)\, \sigma_g = \mathsf{Ex}\; \tau\, \sigma_g$$

$$\mathsf{Ex}\, (\mathbf{or} \left\{ \, \mathbf{for}\; i \in S \colon \tau_i \right)\, \sigma_g = \bigcup_{i \in S} \mathsf{Ex}\, \tau_i\, \sigma_g.$$

The Frame Property

Suppose $\phi \vdash \tau : \langle K, L \rangle$ and

$$\sigma_g \subseteq \sigma'_g \qquad \text{dom } \sigma'_g \perp L \qquad \mathbf{wrong} \notin \mathsf{Ex}\,\tau\,\sigma_g.$$

Then

- $\forall n, \sigma.\ \langle n, \sigma \rangle \in \mathsf{Ex}\,\tau\,\sigma_g$ implies $\sigma \perp (\sigma'_g - \sigma_g)$,
- $\forall n, \sigma.\ \langle n, \sigma \rangle \in \mathsf{Ex}\,\tau\,\sigma_g$ iff $\langle n, \sigma \cup (\sigma'_g - \sigma_g) \rangle \in \mathsf{Ex}\,\tau\,\sigma'_g$,
- $\mathbf{dlock} \in \mathsf{Ex}\,\tau\,\sigma_g$ iff $\mathbf{dlock} \in \mathsf{Ex}\,\tau\,\sigma'_g$.

Formation Rules for Trace Forests

When $\phi, \phi' \in \mathsf{Contexts}$, $\vec{\tau} \in \mathsf{Labels} \rightharpoonup \mathsf{Trees}$, and $\mathsf{dom}\,\phi' \subseteq \mathsf{dom}\,\vec{\tau}$, we write the judgement

$$\phi \vdash \vec{\tau} : \phi'$$

to indicate that $\phi \vdash \vec{\tau}(n) : \phi'(n)$ holds for all $n \in \mathsf{dom}\,\phi'$.

$$\frac{\forall n \in \mathsf{dom}\,\phi'.\ \ \phi \vdash \vec{\tau}(n) : \phi'(n)}{\phi \vdash \vec{\tau} : \phi'} \qquad \frac{\begin{array}{c} n \in \mathsf{dom}\,\phi' \\ \phi \vdash \vec{\tau} : \phi' \end{array}}{\phi \vdash \vec{\tau}(n) : \phi'(n)}$$

Substitution

$$\frac{\begin{array}{c}\phi \vdash \tau : \langle K, L \rangle \\ \phi' \vdash \vec{\tau}' : \phi\end{array}}{\phi' \vdash (\tau / \vec{\tau}') : \langle K, L \rangle}$$

To define this operation, we use an equality judgement on trace trees.

Equality Judgements

When $\phi \vdash \tau : \langle K, L \rangle$ and $\phi \vdash \tau' : \langle K, L \rangle$, the judgement

$$\phi \vdash \underline{\tau = \tau'} : \langle K, L \rangle$$

is well-formed.

Substitution Rules

$$\frac{\phi \vdash \mathbf{halt}\; n : \langle K, L\rangle \qquad \phi' \vdash \vec{\tau}' : \phi}{\phi \vdash \underline{(\mathbf{halt}/\vec{\tau}') = \vec{\tau}'(n)} : \langle K, L\rangle}$$

$$\frac{\phi \vdash \mathbf{dlock} : \langle K, L\rangle \qquad \phi' \vdash \vec{\tau}' : \phi}{\phi \vdash \underline{(\mathbf{dlock}/\vec{\tau}') = \mathbf{dlock}} : \langle K, L\rangle}$$

$$\frac{\phi \vdash \left(\mathbf{start} \left\{ \mathbf{for}\; i \in S\colon \sigma_i\, \tau_i\right) : \langle K, L\rangle \qquad \phi' \vdash \vec{\tau}' : \phi}{\phi \vdash \underline{\left(\left(\mathbf{start} \left\{ \mathbf{for}\; i \in S\colon \sigma_i\, \tau_i\right)\middle/ \vec{\tau}'\right) = \mathbf{start} \left\{ \mathbf{for}\; i \in S\colon \sigma_i\, (\tau_i/\vec{\tau}')\right.} : \langle K, L\rangle}$$

$$\frac{\phi \vdash \mathbf{fin}\; \sigma\, \tau : \langle K, L\rangle \qquad \phi' \vdash \vec{\tau}' : \phi}{\phi \vdash \underline{(\mathbf{fin}\; \sigma\, \tau/\vec{\tau}') = \mathbf{fin}\; \sigma\, (\tau/\vec{\tau}')} : \langle K, L\rangle}$$

More Substitution Rules

$$\frac{\phi \vdash \mathbf{acq}\, k\, \tau : \langle K, L\rangle \qquad \phi' \vdash \vec{\tau}' : \phi}{\phi \vdash \underline{(\mathbf{acq}\, k\, \tau/\vec{\tau}') = \mathbf{acq}\, k\, (\tau/\vec{\tau}')} : \langle K, L\rangle}$$

$$\frac{\phi \vdash \mathbf{rel}\, k\, \tau : \langle K, L\rangle \qquad \phi' \vdash \vec{\tau}' : \phi}{\phi \vdash \underline{(\mathbf{rel}\, k\, \tau/\vec{\tau}') = \mathbf{rel}\, k\, (\tau/\vec{\tau}')} : \langle K, L\rangle}$$

$$\frac{\phi \vdash \left(\mathbf{or}\left\{\ \mathbf{for}\ i \in S\colon \tau_i\right) : \langle K, L\rangle \qquad \phi' \vdash \vec{\tau}' : \phi}{\phi \vdash \underline{\left(\left(\mathbf{or}\left\{\ \mathbf{for}\ i \in S\colon \tau_i\right)\Big/\ \vec{\tau}'\right) = \mathbf{or}\left\{\ \mathbf{for}\ i \in S\colon (\tau_i/\vec{\tau}')\right.} : \langle K, L\rangle}$$

Semantics of Commands

We write $[\![c]\!]_K$ for the trace tree that is the meaning of the command c when, at the beginning of its execution, K is the set of closed locks. Then

$$\frac{}{0\colon \langle K, \{\}\rangle \vdash [\![c]\!]_K : \langle K, \{\}\rangle.}$$

Sequential Composition

$$\frac{}{0\colon \langle K, \{\}\rangle \vdash \underline{[\![c\,;c']\!]_K = ([\![c]\!]_K / 0\colon [\![c']\!]_K)} : \langle K, \{\}\rangle}$$

Interleaving

Let

$$\langle K, L\rangle \perp \langle K', L'\rangle \text{ iff } K \perp K' \text{ and } L \perp L'$$

$$\langle K, L\rangle \cup \langle K', L'\rangle = \langle K \cup K', L \cup L'\rangle$$

Then

$$\frac{\begin{array}{c}\langle K, L\rangle \perp \langle K', L'\rangle \\ \phi \vdash \tau : \langle K, L\rangle \\ \phi' \vdash \tau' : \langle K', L'\rangle\end{array}}{\begin{array}{c}\phi \otimes \phi' \vdash \tau \parallel \tau' : \langle K, L\rangle \cup \langle K', L'\rangle \\ \phi \otimes \phi' \vdash \tau \parallel_l \tau' : \langle K, L\rangle \cup \langle K', L'\rangle \\ \phi \otimes \phi' \vdash \tau \parallel_r \tau' : \langle K, L\rangle \cup \langle K', L'\rangle\end{array}}$$

where

$$\otimes \in \mathsf{Contexts} \times \mathsf{Contexts} \to \mathsf{Contexts}$$

$$\mathsf{dom}(\phi \otimes \phi') = \{\, \langle n, n'\rangle \mid n \in \mathsf{dom}\,\phi, n' \in \mathsf{dom}\,\phi', \phi n \perp \phi n' \,\}$$

$$(\phi \otimes \phi')\langle n, n'\rangle = \phi n \cup \phi n'$$

A Rule for $\|$

$$\frac{\begin{array}{c}\langle K, L\rangle \perp \langle K', L'\rangle \\ \phi \vdash \mathbf{halt}\, n : \langle K, L\rangle \\ \phi' \vdash \mathbf{halt}\, n' : \langle K', L'\rangle\end{array}}{\phi \otimes \phi' \vdash \underline{\mathbf{halt}\, n \parallel \mathbf{halt}\, n' = \mathbf{halt}\langle n, n'\rangle} : \langle K, L\rangle \cup \langle K', L'\rangle}$$

More Rules for $\|$

$$\frac{\langle K, L\rangle \perp \langle K', L'\rangle \quad \phi \vdash \mathbf{acq}\, k\, \tau : \langle K, L\rangle \quad \phi' \vdash \mathbf{fin}\, \sigma'\, \tau' : \langle K', L'\rangle \quad k \notin K'}{\phi \otimes \phi' \vdash \underline{\mathbf{acq}\, k\, \tau \parallel \mathbf{fin}\, \sigma'\, \tau' = \mathbf{or} \left\{ \begin{array}{l} \mathbf{acq}\, k\, \tau \parallel_l \mathbf{fin}\, \sigma'\, \tau' \\ \mathbf{acq}\, k\, \tau \parallel_r \mathbf{fin}\, \sigma'\, \tau' \end{array} \right.} : \langle K, L\rangle \cup \langle K', L'\rangle}$$

$$\frac{\langle K, L\rangle \perp \langle K', L'\rangle \quad \phi \vdash \mathbf{acq}\, k\, \tau : \langle K, L\rangle \quad \phi' \vdash \mathbf{fin}\, \sigma'\, \tau' : \langle K', L'\rangle \quad k \in K'}{\phi \otimes \phi' \vdash \underline{\mathbf{acq}\, k\, \tau \parallel \mathbf{fin}\, \sigma'\, \tau' = \mathbf{acq}\, k\, \tau \parallel_r \mathbf{fin}\, \sigma'\, \tau'} : \langle K, L\rangle \cup \langle K', L'\rangle}$$

Still More Rules for $\|$

$$\frac{\begin{array}{c}\langle K, L\rangle \perp \langle K', L'\rangle \\ \phi \vdash \mathbf{acq}\, k\, \tau : \langle K, L\rangle \\ \phi' \vdash \mathbf{acq}\, k'\, \tau' : \langle K', L'\rangle \\ k \in K' \text{ and } k' \in K\end{array}}{\phi \otimes \phi' \vdash \underline{\mathbf{acq}\, k\, \tau \parallel \mathbf{acq}\, k'\, \tau' = \mathbf{dlock}} : \langle K, L\rangle \cup \langle K', L'\rangle}$$

$$\frac{\begin{array}{c}\langle K, L\rangle \perp \langle K', L'\rangle \\ \phi \vdash \left(\mathbf{or} \begin{cases} \tau_1 \\ \tau_2 \end{cases}\right) : \langle K, L\rangle \\ \phi' \vdash \mathbf{fin}\, \sigma'\, \tau' : \langle K', L'\rangle\end{array}}{\begin{array}{r}\phi \otimes \phi' \vdash \underline{\left(\mathbf{or} \begin{cases} \tau_1 \\ \tau_2 \end{cases}\right) \parallel \mathbf{fin}\, \sigma'\, \tau' = \left(\mathbf{or} \begin{cases} \tau_1 \\ \tau_2 \end{cases}\right) \parallel_l \mathbf{fin}\, \sigma'\, \tau'} \\ : \langle K, L\rangle \cup \langle K', L'\rangle\end{array}}$$

The General Case

$$\frac{\begin{array}{c}\langle K, L\rangle \perp \langle K', L'\rangle \\ \phi \vdash \tau_1 : \langle K, L\rangle \\ \phi' \vdash \tau_2 : \langle K', L'\rangle \\ \text{auxilliary premisses}\end{array}}{\phi \otimes \phi' \vdash \underline{\tau_1 \parallel \tau_2 = \tau_3} : \langle K, L\rangle \cup \langle K', L'\rangle}$$

Here τ_1, τ_2, and τ_3 are metavariables whose values are specified on the next slide. In some cases, τ_1 or τ_2 will have associated auxilliary predicates that must be added as premisses to the instance of the rule.

$\tau_1 =$ $\tau_2 =$	**halt** n	**dlock** **acq** $k\,\tau$ $k \in K'$	**acq** **rel** **start** **fin** $k\,\tau$ $k\,\tau$ $\cdots$ $\sigma\,\tau$ $k \notin K'$	**or** $\cdots$
halt n'	**halt** $\langle n, n' \rangle$	**dlock**	$\tau_1 \parallel_l \tau_2$	$\tau_1 \parallel_l \tau_2$
dlock **acq** $k'\,\tau'$ $k' \in K$	**dlock**	**dlock**	$\tau_1 \parallel_l \tau_2$	$\tau_1 \parallel_l \tau_2$
acq $k'\,\tau'$ $k' \notin K$ **rel** $k'\,\tau'$ **start** $\cdots$ **fin** $\sigma'\,\tau'$	$\tau_1 \parallel_r \tau_2$	$\tau_1 \parallel_r \tau_2$	**or** $\left\{ \begin{array}{l} \tau_1 \parallel_l \tau_2 \\ \tau_1 \parallel_r \tau_2 \end{array} \right.$	$\tau_1 \parallel_l \tau 2$
or $\cdots$	$\tau_1 \parallel_r \tau_2$	$\tau_1 \parallel_r \tau_2$	$\tau_1 \parallel_r \tau_2$	$\tau_1 \parallel_l \tau_2$

Rules for $\|_l$

$$\frac{\begin{array}{c}\langle K, L\rangle \perp \langle K', L'\rangle \\ \phi \vdash \mathbf{acq}\, k\, \tau : \langle K, L\rangle \\ \phi' \vdash \tau' : \langle K', L'\rangle \\ k \notin K'\end{array}}{\phi \otimes \phi' \vdash \underline{(\mathbf{acq}\, k\, \tau) \|_l \tau' = \mathbf{acq}\, k\, (\tau \parallel \tau')} : \langle K, L\rangle \cup \langle K', L'\rangle}$$

$$\frac{\begin{array}{c}\langle K, L\rangle \perp \langle K', L'\rangle \\ \phi \vdash \mathbf{rel}\, k\, \tau : \langle K, L\rangle \\ \phi' \vdash \tau' : \langle K', L'\rangle\end{array}}{\phi \otimes \phi' \vdash \underline{(\mathbf{rel}\, k\, \tau) \|_l \tau' = \mathbf{rel}\, k\, (\tau \parallel \tau')} : \langle K, L\rangle \cup \langle K', L'\rangle}$$

$$\frac{\begin{array}{c}\langle K, L\rangle \perp \langle K', L'\rangle \\ \phi \vdash \mathbf{fin}\, \sigma\, \tau : \langle K, L\rangle \\ \phi' \vdash \tau' : \langle K', L'\rangle\end{array}}{\phi \otimes \phi' \vdash \underline{(\mathbf{fin}\, \sigma\, \tau) \|_l \tau' = \mathbf{fin}\, \sigma\, (\tau \parallel \tau')} : \langle K, L\rangle \cup \langle K', L'\rangle}$$

$$\frac{\begin{array}{c}\langle K, L\rangle \perp \langle K', L'\rangle \\ \phi \vdash \mathbf{or} \left\{\ \mathbf{for}\ i \in S \colon \tau_i \right. : \langle K, L\rangle \\ \phi' \vdash \tau' : \langle K', L'\rangle\end{array}}{\begin{array}{r}\phi \otimes \phi' \vdash \underline{\left(\mathbf{or} \left\{\ \mathbf{for}\ i \in S \colon \tau_i \right.\right) \|_l \tau' = \mathbf{or} \left\{\ \mathbf{for}\ i \in S \colon (\tau_i \parallel \tau') \right.} \\ : \langle K, L\rangle \cup \langle K', L'\rangle\end{array}}$$

Another Rule for $\|_l$

$$\frac{\begin{array}{c}\langle K, L\rangle \perp \langle K', L'\rangle \\ \phi \vdash \mathbf{start}\left\{\, \mathbf{for}\; i \in S \colon \sigma_i\, \tau_i : \langle K, L\rangle \right. \\ \phi' \vdash \tau' : \langle K', L'\rangle\end{array}}{\begin{array}{l}\phi \otimes \phi' \vdash \underline{\left(\mathbf{start}\left\{\, \mathbf{for}\; i \in S \colon \sigma_i\, \tau_i\right.\right) \|_l\, \tau' =} \\ \qquad\qquad \underline{\mathbf{start}\left\{\, \mathbf{for}\; i \in S' \colon \sigma_i\,(\tau_i \parallel \tau')\right.} : \langle K, L\rangle \cup \langle K', L'\rangle\end{array}}$$

where

$$S' = \{\, i \in S \mid \mathsf{dom}\ \sigma_i \perp L' \,\}.$$

Semantics of Concurrent Composition

$$\frac{}{0\colon \langle K \cup K', \{\}\rangle \vdash \underline{\llbracket c \parallel c' \rrbracket_{K \cup K'} = ((\llbracket c \rrbracket_K \parallel \llbracket c' \rrbracket_{K'})/\langle 0, 0\rangle\colon \mathbf{halt}\ 0)} : \langle K \cup K', \{\}\rangle}$$

$$\frac{k \notin K}{0\colon \langle K, \{\}\rangle \vdash \underline{\llbracket \mathbf{with}\ k\ \mathbf{do}\ c \rrbracket_K = \mathbf{acq}\ k\ (\llbracket c \rrbracket_{K \cup \{k\}}/0\colon \mathbf{rel}\ k\ \mathbf{halt}\ 0)} : \langle K, \{\}\rangle}$$

$$\frac{k \in K}{0\colon \langle K, \{\}\rangle \vdash \underline{\llbracket \mathbf{with}\ k\ \mathbf{do}\ c \rrbracket_K = \mathbf{dlock}} : \langle K, \{\}\rangle}$$

Equivalence of Trace Trees

$$\tau_1 \cong \tau_2 \text{ iff } \forall G \in \mathsf{Contexts}.\ \forall \sigma_g \in \mathsf{States}.$$
$$\mathsf{Ex}\,(\llbracket G \rrbracket\, \tau_1)\, \sigma_g = \mathsf{Ex}\,(\llbracket G \rrbracket\, \tau_2)\, \sigma_g$$

where a context is a command with a hole in it, and $\llbracket G \rrbracket \llbracket c \rrbracket = \llbracket G[c] \rrbracket$.

$$\tau_1 \simeq \tau_2 \text{ iff } \forall \tau'.\ \forall \sigma_g \in \mathsf{States}.$$
$$\mathsf{Ex}\,(\tau_1 \parallel \tau')\, \sigma_g = \mathsf{Ex}\,(\tau_2 \parallel \tau')\, \sigma_g.$$

We conjecture that

$$\tau_1 \simeq \tau_2 \text{ implies } \tau_1 \cong \tau_2.$$

Rewrite Rules (Speculative)

$$\textbf{start}\left\{\ \textbf{for}\ i \in S\colon\ \sigma_i\ \textbf{start}\left\{\ \textbf{for}\ j \in S_i\colon\ \sigma_{ij}\ \tau_{ij}\right.\right.$$

$$\Rightarrow \textbf{start}\left\{\ \textbf{for}\ i \in S, j \in S_i\ \textbf{s.t.}\ \sigma_i \perp \sigma_{ij}\colon\ (\sigma_i \cup \sigma_{ij})\ \tau_{ij}\right.$$

$$\textbf{fin}\ \sigma\ \textbf{start}\left\{\ \textbf{for}\ i \in S\colon\ \sigma_i\ \tau_i\right.$$

$$\Rightarrow \textbf{start}\left\{\ \textbf{for}\ i \in S\ \textbf{s.t.}\ \sigma_i \smile \sigma\colon\ (\sigma_i - \sigma)\ \textbf{fin}\,(\sigma - \sigma_i)\ \tau_i\right.$$

$$\textbf{fin}\ \sigma\ \textbf{fin}\ \sigma'\ \tau\ \Rightarrow\ \textbf{fin}\ (\sigma \cup \sigma')\ \tau \quad \text{when } \sigma \perp \sigma'$$

$$\tau \Rightarrow \textbf{start}\left\{\ [\,]\ \tau\right. \qquad\qquad \tau \Rightarrow \textbf{fin}\,[\,]\ \tau$$

Rewrite Rules for Lock Operations

$$\textbf{rel}\ k\ \textbf{start}\left\{\ \textbf{for}\ i \in S\text{:}\ \sigma_i\ \tau_i\right.$$

$$\Rightarrow \textbf{start}\left\{\ \textbf{for}\ i \in S\text{:}\ \sigma_i\ \textbf{rel}\ k\ \tau_i\right.$$

$$\textbf{fin}\ \sigma\ \textbf{acq}\ k\ \tau\ \ \Rightarrow\ \ \textbf{acq}\ k\ \textbf{fin}\ \sigma\ \tau$$

$$\textbf{rel}\ \tau\ \ \Rightarrow\ \ \textbf{fin}[\,]\ \textbf{rel}\ \tau$$

Rewrite Rules for Nondeterminism

$$\textbf{or}\begin{cases}\textbf{start}\left\{\ \textbf{for } i \in S\colon\ \sigma_i\ \tau_i\right.\\ \textbf{start}\left\{\ \textbf{for } j \in S'\colon\ \sigma'_j\ \tau'_j\right.\end{cases}$$

$$\Rightarrow \textbf{start}\left\{\ \textbf{for } i \in S, j \in S' \textbf{ s.t. } \sigma_i \smile \sigma'_j\colon\ (\sigma_i \cup \sigma'_j)\ \textbf{or}\begin{cases}\textbf{fin}\ (\sigma'_j - \sigma_i)\ \tau_i\\ \textbf{fin}\ (\sigma_i - \sigma'_j)\ \tau'_j\end{cases}\right.$$

$$\textbf{fin}\ \textbf{or}\begin{cases}\tau_1\\ \tau_2\\ \vdots\end{cases} \Rightarrow \textbf{or}\begin{cases}\textbf{fin}\ \tau_1\\ \textbf{fin}\ \tau_2\\ \vdots\end{cases} \qquad \textbf{rel}\ \textbf{or}\begin{cases}\tau_1\\ \tau_2\\ \vdots\end{cases} \Rightarrow \textbf{or}\begin{cases}\textbf{rel}\ \tau_1\\ \textbf{rel}\ \tau_2\\ \vdots\end{cases}$$

$$\tau \Rightarrow \textbf{or}\left\{\ \tau\right.$$

A Conjectured Normal Form

$$\langle \mathsf{tree} \rangle ::= \mathbf{start} \begin{cases} \langle \mathsf{state} \rangle \; \mathbf{or} \begin{cases} \langle \mathsf{midtree} \rangle \\ \langle \mathsf{midtree} \rangle \\ \quad \vdots \end{cases} \\ \langle \mathsf{state} \rangle \; \mathbf{or} \begin{cases} \langle \mathsf{midtree} \rangle \\ \langle \mathsf{midtree} \rangle \\ \quad \vdots \end{cases} \\ \quad \vdots \end{cases}$$

A Conjectured Normal Form (continued)

$$\langle \mathsf{midtree} \rangle ::= \mathbf{acq}\,\langle \mathsf{lock} \rangle\,\mathbf{start} \begin{cases} \langle \mathsf{state} \rangle\,\mathbf{or} \begin{cases} \langle \mathsf{midtree} \rangle \\ \langle \mathsf{midtree} \rangle \\ \quad \vdots \end{cases} \\ \langle \mathsf{state} \rangle\,\mathbf{or} \begin{cases} \langle \mathsf{midtree} \rangle \\ \langle \mathsf{midtree} \rangle \\ \quad \vdots \end{cases} \\ \quad \vdots \end{cases}$$

$$\mid \mathbf{fin}\,\langle \mathsf{state} \rangle\,\mathbf{rel}\,\langle \mathsf{lock} \rangle\ \langle \mathsf{midtree} \rangle$$

$$\mid \langle \mathsf{endtree} \rangle$$

$$\langle \mathsf{endtree} \rangle ::= \mathbf{fin}\,\langle \mathsf{state} \rangle\,\mathbf{halt}\,\langle \mathsf{label} \rangle \mid \mathbf{dlock}$$

Future Directions

- Allow nontermination
- Allow unbounded nondeterminism
- Introduce allocation and disposal.
- Introduce passivity (i.e., allow read-only interference).
- Use to model separation logic for shared-variable concurrency.

A Rewriting Sequence (1)

$$\mathbf{fin}\ \sigma\ \mathbf{start}\left\{\ \mathbf{for}\ i \in S\colon\ \sigma_i\ \tau_i\right.$$
$$\Rightarrow \mathbf{start}\left\{\ \mathbf{for}\ i \in S\ \mathbf{s.t.}\ \sigma_i \smile \sigma\colon\ (\sigma_i - \sigma)\ \mathbf{fin}\,(\sigma - \sigma_i)\ \tau_i\right.$$

$$[\![[\mathsf{x}] := [\mathsf{x}] + 1\ ;\ [\mathsf{y}] := [\mathsf{y}] + 2]\!]$$
$$= \mathbf{start}\left\{\ \mathbf{for}\ m, m' \in \mathcal{Z}\colon\ [\,\mathsf{x}\colon m \mid m\colon m'\,]\right.$$
$$\mathbf{fin}\,[\,\mathsf{x}\colon m \mid m\colon m' + 1\,]$$
$$\mathbf{start}\left\{\ \mathbf{for}\ n, n' \in \mathcal{Z}\colon\ [\,\mathsf{y}\colon n \mid n\colon n'\,]\right.$$
$$\mathbf{fin}\,[\,\mathsf{y}\colon n \mid n\colon n' + 2\,]\ \mathbf{halt}\ 0$$
$$\Rightarrow \mathbf{start}\left\{\ \mathbf{for}\ m, m' \in \mathcal{Z}\colon\ [\,\mathsf{x}\colon m \mid m\colon m'\,]\right.$$
$$\mathbf{start}\left\{\ \mathbf{for}\ n, n' \in \mathcal{Z}\ \mathbf{s.t.}\ [\,\mathsf{y}\colon n \mid n\colon n'\,] \smile [\,\mathsf{x}\colon m \mid m\colon m' + 1\,]\colon\right.$$
$$([\,\mathsf{y}\colon n \mid n\colon n'\,] - [\,\mathsf{x}\colon m \mid m\colon m' + 1\,])$$
$$\mathbf{fin}\,([\,\mathsf{x}\colon m \mid m\colon m' + 1\,] - [\,\mathsf{y}\colon n \mid n\colon n'\,])$$
$$\mathbf{fin}\,[\,\mathsf{y}\colon n \mid n\colon n' + 2\,]\ \mathbf{halt}\ 0$$
$$= \mathbf{start}\left\{\ \mathbf{for}\ m, m' \in \mathcal{Z}\colon\ [\,\mathsf{x}\colon m \mid m\colon m'\,]\right.$$
$$\mathbf{start}\left\{\begin{array}{l} \mathbf{for}\ n, n' \in \mathcal{Z}\ \mathbf{s.t.}\ n \neq m\colon\ [\,\mathsf{y}\colon n \mid n\colon n'\,] \\ \quad \mathbf{fin}\,[\,\mathsf{x}\colon m \mid m\colon m' + 1\,]\ \mathbf{fin}\,[\,\mathsf{y}\colon n \mid n\colon n' + 2\,]\ \mathbf{halt}\ 0 \\ [\,\mathsf{y}\colon m\,]\ \mathbf{fin}\,[\,\mathsf{x}\colon m\,]\ \mathbf{fin}\,[\,\mathsf{y}\colon m \mid m\colon m' + 3\,]\ \mathbf{halt}\ 0 \end{array}\right.$$

A Rewriting Sequence (1) (continued)

$$\mathbf{fin}\ \sigma\ \mathbf{fin}\ \sigma'\ \tau \Rightarrow \mathbf{fin}\ (\sigma \cup \sigma')\ \tau \quad \text{when } \sigma \perp \sigma'$$

$$\mathbf{start}\left\{\ \mathbf{for}\ m, m' \in \mathcal{Z}\text{:}\ [\,\mathsf{x}\text{:}\, m \mid m\text{:}\, m'\,]\right.$$

$$\mathbf{start}\left\{\begin{array}{l} \mathbf{for}\ n, n' \in \mathcal{Z}\ \mathbf{s.t.}\ n \neq m\text{:}\ [\,\mathsf{y}\text{:}\, n \mid n\text{:}\, n'\,] \\ \quad \mathbf{fin}\,[\,\mathsf{x}\text{:}\, m \mid m\text{:}\, m' + 1\,]\ \mathbf{fin}\,[\,\mathsf{y}\text{:}\, n \mid n\text{:}\, n' + 2\,]\ \mathbf{halt}\ 0 \\ [\,\mathsf{y}\text{:}\, m\,]\ \mathbf{fin}\,[\,\mathsf{x}\text{:}\, m\,]\ \mathbf{fin}\,[\,\mathsf{y}\text{:}\, m \mid m\text{:}\, m' + 3\,]\ \mathbf{halt}\ 0 \end{array}\right.$$

$$\Rightarrow\ \mathbf{start}\left\{\ \mathbf{for}\ m, m' \in \mathcal{Z}\text{:}\ [\,\mathsf{x}\text{:}\, m \mid m\text{:}\, m'\,]\right.$$

$$\mathbf{start}\left\{\begin{array}{l} \mathbf{for}\ n, n' \in \mathcal{Z}\ \mathbf{s.t.}\ n \neq m\text{:}\ [\,\mathsf{y}\text{:}\, n \mid n\text{:}\, n'\,] \\ \quad \mathbf{fin}\,[\,\mathsf{x}\text{:}\, m \mid m\text{:}\, m' + 1 \mid \mathsf{y}\text{:}\, n \mid n\text{:}\, n' + 2\,]\ \mathbf{halt}\ 0 \\ [\,\mathsf{y}\text{:}\, m\,]\ \mathbf{fin}\,[\,\mathsf{x}\text{:}\, m \mid \mathsf{y}\text{:}\, m \mid m\text{:}\, m' + 3\,]\ \mathbf{halt}\ 0 \end{array}\right.$$

A Rewriting Sequence (1) (continued)

$$\textbf{start}\left\{\ \textbf{for}\ i \in S\colon\ \sigma_i\ \textbf{start}\left\{\ \textbf{for}\ j \in S_i\colon\ \sigma_{ij}\ \tau_{ij}\right.\right.$$

$$\Rightarrow \textbf{start}\left\{\ \textbf{for}\ i \in S, j \in S_i\ \textbf{s.t.}\ \sigma_i \perp \sigma_{ij}\colon\ (\sigma_i \cup \sigma_{ij})\ \tau_{ij}\right.$$

$$\begin{array}{l}\textbf{start}\left\{\ \textbf{for}\ m, m' \in \mathcal{Z}\colon\ [\,\mathsf{x}\colon m \mid m\colon m'\,]\right. \\ \textbf{start}\left\{\begin{array}{l}\textbf{for}\ n, n' \in \mathcal{Z}\ \textbf{s.t.}\ n \neq m\colon\ [\,\mathsf{y}\colon n \mid n\colon n'\,] \\ \quad \textbf{fin}\,[\,\mathsf{x}\colon m \mid m\colon m'+1 \mid \mathsf{y}\colon n \mid n\colon n'+2\,]\ \textbf{halt}\ 0 \\ [\,\mathsf{y}\colon m\,]\ \textbf{fin}\,[\,\mathsf{x}\colon m \mid \mathsf{y}\colon m \mid m\colon m'+3\,]\ \textbf{halt}\ 0\end{array}\right.\end{array}$$

$$\Rightarrow\ \textbf{start}\left\{\begin{array}{l}\textbf{for}\ m, m', n, n' \in \mathcal{Z}\ \textbf{s.t.}\ n \neq m\colon \\ \qquad [\,\mathsf{x}\colon m \mid m\colon m' \mid \mathsf{y}\colon n \mid n\colon n'\,] \\ \quad \textbf{fin}\,[\,\mathsf{x}\colon m \mid m\colon m'+1 \mid \mathsf{y}\colon n \mid n\colon n'+2\,]\ \textbf{halt}\ 0 \\ \textbf{for}\ m, m' \in \mathcal{Z}\colon\ [\,\mathsf{x}\colon m \mid \mathsf{y}\colon m \mid m\colon m'\,] \\ \quad \textbf{fin}\,[\,\mathsf{x}\colon m \mid \mathsf{y}\colon m \mid m\colon m'+3\,]\ \textbf{halt}\ 0\end{array}\right.$$

A Rewriting Sequence (2)

$$\textbf{or}\begin{cases}\textbf{start}\left\{\ \textbf{for}\ i \in S\colon\ \sigma_i\ \tau_i\right.\\ \textbf{start}\left\{\ \textbf{for}\ j \in S'\colon\ \sigma'_j\ \tau'_j\right.\end{cases}$$

$$\Rightarrow \textbf{start}\left\{\ \textbf{for}\ i \in S, j \in S'\ \textbf{s.t.}\ \sigma_i \smile \sigma'_j\colon\ (\sigma_i \cup \sigma'_j)\ \textbf{or}\begin{cases}\textbf{fin}\ (\sigma'_j - \sigma_i)\ \tau_i\\ \textbf{fin}\ (\sigma_i - \sigma'_j)\ \tau'_j\end{cases}\right.$$

$$[\![\mathsf{x} := 0\ \textbf{or}\ \mathsf{y} := 0]\!]$$

$$= \ \textbf{or}\begin{cases}\textbf{start}\left\{\ \textbf{for}\ m \in \mathcal{Z}\colon\ [\,\mathsf{x}\colon m\,]\ \textbf{fin}\ [\,\mathsf{x}\colon 0\,]\ \textbf{halt}\ 0\right.\\ \textbf{start}\left\{\ \textbf{for}\ n \in \mathcal{Z}\colon\ [\,\mathsf{y}\colon n\,]\ \textbf{fin}\ [\,\mathsf{y}\colon 0\,]\ \textbf{halt}\ 0\right.\end{cases}$$

$$\Rightarrow\ \textbf{start}\left\{\ \textbf{for}\ m, n \in \mathcal{Z}\colon\ [\,\mathsf{x}\colon m \mid \mathsf{y}\colon n\,]\ \textbf{or}\begin{cases}\textbf{fin}\ [\,\mathsf{y}\colon n\,]\ \textbf{fin}\ [\,\mathsf{x}\colon 0\,]\ \textbf{halt}\ 0\\ \textbf{fin}\ [\,\mathsf{x}\colon m\,]\ \textbf{fin}\ [\,\mathsf{y}\colon 0\,]\ \textbf{halt}\ 0\end{cases}\right.$$

$$\Rightarrow\ \textbf{start}\left\{\ \textbf{for}\ m, n \in \mathcal{Z}\colon\ [\,\mathsf{x}\colon m \mid \mathsf{y}\colon n\,]\ \textbf{or}\begin{cases}\textbf{fin}\ [\,\mathsf{x}\colon 0 \mid \mathsf{y}\colon n\,]\ \textbf{halt}\ 0\\ \textbf{fin}\ [\,\mathsf{x}\colon m \mid \mathsf{y}\colon 0\,]\ \textbf{halt}\ 0\end{cases}\right.$$

A Rewriting Sequence (3)

$$\textbf{or}\begin{cases}\textbf{start}\left\{\textbf{for } i\in S\colon\ \sigma_i\ \tau_i\right.\\ \textbf{start}\left\{\textbf{for } j\in S'\colon\ \sigma'_j\ \tau'_j\right.\end{cases}$$

$$\Rightarrow \textbf{start}\left\{\textbf{for } i\in S, j\in S' \textbf{ s.t. } \sigma_i \smile \sigma'_j\colon\ (\sigma_i\cup\sigma'_j)\ \textbf{or}\begin{cases}\textbf{fin}\ (\sigma'_j-\sigma_i)\ \tau_i\\ \textbf{fin}\ (\sigma_i-\sigma'_j)\ \tau'_j\end{cases}\right.$$

$$[\![(\mathsf{x} := 0\ ;\ \mathsf{y} := \mathsf{y})\ \textbf{or}\ (\mathsf{y} := 0\ ;\ \mathsf{x} := \mathsf{x})]\!]$$

$$= \textbf{or}\begin{cases}\textbf{start}\left\{\textbf{for } m,n\in\mathcal{Z}\colon [\mathsf{x}\colon m \mid \mathsf{y}\colon n]\ \textbf{fin}\ [\mathsf{x}\colon 0 \mid \mathsf{y}\colon n]\ \textbf{halt}\ 0\right.\\ \textbf{start}\left\{\textbf{for } m',n'\in\mathcal{Z}\colon [\mathsf{x}\colon m' \mid \mathsf{y}\colon n']\ \textbf{fin}\ [\mathsf{x}\colon m' \mid \mathsf{y}\colon 0]\ \textbf{halt}\ 0\right.\end{cases}$$

$$\Rightarrow \textbf{start}\left\{\begin{array}{l}\textbf{for } m,n,m',n'\in\mathcal{Z} \textbf{ s.t. } [\mathsf{x}\colon m \mid \mathsf{y}\colon n]\smile[\mathsf{x}\colon m' \mid \mathsf{y}\colon n']\colon\\ \quad([\mathsf{x}\colon m \mid \mathsf{y}\colon n]\cup[\mathsf{x}\colon m' \mid \mathsf{y}\colon n'])\\ \quad\textbf{or}\begin{cases}\textbf{fin}\ ([\mathsf{x}\colon m' \mid \mathsf{y}\colon n'] - [\mathsf{x}\colon m \mid \mathsf{y}\colon n])\ \textbf{fin}\ [\mathsf{x}\colon 0 \mid \mathsf{y}\colon n]\ \textbf{halt}\ 0\\ \textbf{fin}\ ([\mathsf{x}\colon m \mid \mathsf{y}\colon n] - [\mathsf{x}\colon m' \mid \mathsf{y}\colon n'])\ \textbf{fin}\ [\mathsf{x}\colon m \mid \mathsf{y}\colon 0]\ \textbf{halt}\ 0\end{cases}\end{array}\right.$$

$$\Rightarrow \textbf{start}\left\{\begin{array}{l}\textbf{for } m,n\in\mathcal{Z}\colon\ [\mathsf{x}\colon m \mid \mathsf{y}\colon n]\\ \quad\textbf{or}\begin{cases}\textbf{fin}\ [\,]\ \textbf{fin}\ [\mathsf{x}\colon 0 \mid \mathsf{y}\colon n]\ \textbf{halt}\ 0\\ \textbf{fin}\ [\,]\ \textbf{fin}\ [\mathsf{x}\colon m \mid \mathsf{y}\colon 0]\ \textbf{halt}\ 0\end{cases}\end{array}\right.$$

$$\Rightarrow \textbf{start}\left\{\begin{array}{l}\textbf{for } m,n\in\mathcal{Z}\colon\ [\mathsf{x}\colon m \mid \mathsf{y}\colon n]\\ \quad\textbf{or}\begin{cases}\textbf{fin}\ [\mathsf{x}\colon 0 \mid \mathsf{y}\colon n]\ \textbf{halt}\ 0\\ \textbf{fin}\ [\mathsf{x}\colon m \mid \mathsf{y}\colon 0]\ \textbf{halt}\ 0\end{cases}\end{array}\right.$$

A Rewriting Sequence (4)

$$\mathbf{fin}\ \sigma\ \mathbf{acq}\ k\ \tau \ \Rightarrow\ \mathbf{acq}\ k\ \mathbf{fin}\ \sigma\ \tau$$

$$[\![\mathsf{x} := \mathsf{x} \times \mathsf{x}\,;\ \mathbf{with}\ \mathsf{r}\ \mathbf{do}\ (\mathsf{x} := \mathsf{x} + 1\,;\ \mathsf{y} := 0)]\!]$$

$$= \ \mathbf{start}\Big\{\ \mathbf{for}\ i \in \mathcal{Z}\colon\ [\,\mathsf{x}\colon i\,]\ \mathbf{fin}\,[\,\mathsf{x}\colon i \times i\,]\ \mathbf{acq}\ \mathsf{r}$$
$$\mathbf{start}\Big\{\ \mathbf{for}\ j, k \in \mathcal{Z}\colon$$
$$[\,\mathsf{x}\colon j \mid \mathsf{y}\colon k\,]\ \mathbf{fin}\ [\,\mathsf{x}\colon j + 1 \mid \mathsf{y}\colon 0\,]\ \mathbf{rel}\ \mathsf{r}\ \mathbf{halt}\ 0$$

$$\Rightarrow \ \mathbf{start}\Big\{\ \mathbf{for}\ i \in \mathcal{Z}\colon\ [\,\mathsf{x}\colon i\,]\ \mathbf{acq}\ \mathsf{r}$$
$$\mathbf{fin}\,[\,\mathsf{x}\colon i \times i\,]\ \mathbf{start}\Big\{\ \mathbf{for}\ j, k \in \mathcal{Z}\colon$$
$$[\,\mathsf{x}\colon j \mid \mathsf{y}\colon k\,]\ \mathbf{fin}\ [\,\mathsf{x}\colon j + 1 \mid \mathsf{y}\colon 0\,]\ \mathbf{rel}\ \mathsf{r}\ \mathbf{halt}\ 0$$

A Rewriting Sequence (4) (continued)

$$\textbf{fin}\ \sigma\ \textbf{start}\left\{\ \textbf{for}\ i \in S\colon\ \sigma_i\ \tau_i\right.$$

$$\Rightarrow \textbf{start}\left\{\ \textbf{for}\ i \in S\ \textbf{s.t.}\ \sigma_i \smile \sigma\colon\ (\sigma_i - \sigma)\ \textbf{fin}\,(\sigma - \sigma_i)\ \tau_i\right.$$

$$\textbf{start}\left\{\ \textbf{for}\ i \in \mathcal{Z}\colon\ [\,\mathsf{x}\colon i\,]\ \textbf{acq}\ \mathsf{r}\right.$$
$$\quad \textbf{fin}\,[\,\mathsf{x}\colon i \times i\,]\ \textbf{start}\left\{\ \textbf{for}\ j, k \in \mathcal{Z}\colon\right.$$
$$\qquad [\,\mathsf{x}\colon j \mid \mathsf{y}\colon k\,]\ \textbf{fin}\ [\,\mathsf{x}\colon j + 1 \mid \mathsf{y}\colon 0\,]\ \textbf{rel}\ \mathsf{r}\ \textbf{halt}\ 0$$

$$\Rightarrow\ \textbf{start}\left\{\ \textbf{for}\ i \in \mathcal{Z}\colon\ [\,\mathsf{x}\colon i\,]\ \textbf{acq}\ \mathsf{r}\right.$$
$$\quad \textbf{start}\left\{\ \textbf{for}\ j, k \in \mathcal{Z}\ \textbf{s.t.}\ [\,\mathsf{x}\colon j \mid \mathsf{y}\colon k\,] \smile [\,\mathsf{x}\colon i \times i\,]\colon\right.$$
$$\qquad ([\,\mathsf{x}\colon j \mid \mathsf{y}\colon k\,] - [\,\mathsf{x}\colon i \times i\,])\ \textbf{fin}\ ([\,\mathsf{x}\colon i \times i\,] - [\,\mathsf{x}\colon j \mid \mathsf{y}\colon k\,])$$
$$\qquad\quad \textbf{fin}\ [\,\mathsf{x}\colon j + 1 \mid \mathsf{y}\colon 0\,]\ \textbf{rel}\ \mathsf{r}\ \textbf{halt}\ 0$$

$$\Rightarrow\ \textbf{start}\left\{\ \textbf{for}\ i \in \mathcal{Z}\colon\ [\,\mathsf{x}\colon i\,]\ \textbf{acq}\ \mathsf{r}\right.$$
$$\quad \textbf{start}\left\{\ \textbf{for}\ k \in \mathcal{Z}\colon\right.$$
$$\qquad [\,\mathsf{y}\colon k\,]\ \textbf{fin}\ [\,]$$
$$\qquad\quad \textbf{fin}\ [\,\mathsf{x}\colon i \times i + 1 \mid \mathsf{y}\colon 0\,]\ \textbf{rel}\ \mathsf{r}\ \textbf{halt}\ 0$$

$$\Rightarrow\ \textbf{start}\left\{\ \textbf{for}\ i \in \mathcal{Z}\colon\ [\,\mathsf{x}\colon i\,]\ \textbf{acq}\ \mathsf{r}\right.$$
$$\quad \textbf{start}\left\{\ \textbf{for}\ k \in \mathcal{Z}\colon\right.$$
$$\qquad [\,\mathsf{y}\colon k\,]\ \textbf{fin}\ [\,\mathsf{x}\colon i \times i + 1 \mid \mathsf{y}\colon 0\,]\ \textbf{rel}\ \mathsf{r}\ \textbf{halt}\ 0$$

A Rewriting Sequence (5)

$$\mathbf{fin}\ \sigma\ \mathbf{acq}\ k\ \tau\ \ \Rightarrow\ \ \mathbf{acq}\ k\ \mathbf{fin}\ \sigma\ \tau$$

$$[\![\mathsf{x} := \mathsf{x} \times \mathsf{x} + 1\ ;\ \mathbf{with}\ \mathsf{r}\ \mathbf{do}\ \mathsf{y} := 0]\!]$$

$$\begin{aligned}
= \quad & \mathbf{start} \big\{ \mathbf{for}\ i \in \mathcal{Z}\colon\ [\,\mathsf{x}\colon i\,]\ \mathbf{fin}\,[\,\mathsf{x}\colon i \times i + 1\,]\ \mathbf{acq}\ \mathsf{r} \\
& \quad \mathbf{start} \big\{ \mathbf{for}\ k \in \mathcal{Z}\colon \\
& \qquad [\,\mathsf{y}\colon k\,]\ \mathbf{fin}\ [\,\mathsf{y}\colon 0\,]\ \mathbf{rel}\ \mathsf{r}\ \mathbf{halt}\ 0 \\
\Rightarrow \quad & \mathbf{start} \big\{ \mathbf{for}\ i \in \mathcal{Z}\colon\ [\,\mathsf{x}\colon i\,]\ \mathbf{acq}\ \mathsf{r} \\
& \quad \mathbf{fin}\,[\,\mathsf{x}\colon i \times i + 1\,]\ \mathbf{start} \big\{ \mathbf{for}\ k \in \mathcal{Z}\colon \\
& \qquad [\,\mathsf{y}\colon k\,]\ \mathbf{fin}\ [\,\mathsf{y}\colon 0\,]\ \mathbf{rel}\ \mathsf{r}\ \mathbf{halt}\ 0
\end{aligned}$$

A Rewriting Sequence (5) (continued)

$$\textbf{fin}\ \sigma\ \textbf{start}\left\{\ \textbf{for}\ i \in S\text{:}\ \sigma_i\ \tau_i\right.$$
$$\Rightarrow \textbf{start}\left\{\ \textbf{for}\ i \in S\ \textbf{s.t.}\ \sigma_i \smile \sigma\text{:}\ (\sigma_i - \sigma)\ \textbf{fin}\,(\sigma - \sigma_i)\ \tau_i\right.$$

$$\textbf{start}\left\{\ \textbf{for}\ i \in \mathcal{Z}\text{:}\ [\,\mathsf{x}\text{:}\ i\,]\,\textbf{acq}\ \mathsf{r}\right.$$
$$\textbf{fin}\,[\,\mathsf{x}\text{:}\ i \times i + 1\,]\ \textbf{start}\left\{\ \textbf{for}\ j, k \in \mathcal{Z}\text{:}\right.$$
$$[\,\mathsf{y}\text{:}\ k\,]\ \textbf{fin}\ [\,\mathsf{y}\text{:}\ 0\,]\ \textbf{rel}\ \mathsf{r}\ \textbf{halt}\ 0$$

$$\Rightarrow\ \textbf{start}\left\{\ \textbf{for}\ i \in \mathcal{Z}\text{:}\ [\,\mathsf{x}\text{:}\ i\,]\,\textbf{acq}\ \mathsf{r}\right.$$
$$\textbf{start}\left\{\ \textbf{for}\ k \in \mathcal{Z}\ \textbf{s.t.}\ [\,\mathsf{y}\text{:}\ k\,] \smile [\,\mathsf{x}\text{:}\ i \times i + 1\,]\text{:}\right.$$
$$([\,\mathsf{y}\text{:}\ k\,] - [\,\mathsf{x}\text{:}\ i \times i + 1\,])\ \textbf{fin}\ ([\,\mathsf{x}\text{:}\ i \times i + 1\,] - [\,\mathsf{y}\text{:}\ k\,])$$
$$\textbf{fin}\ [\,\mathsf{y}\text{:}\ 0\,]\ \textbf{rel}\ \mathsf{r}\ \textbf{halt}\ 0$$

$$\Rightarrow\ \textbf{start}\left\{\ \textbf{for}\ i \in \mathcal{Z}\text{:}\ [\,\mathsf{x}\text{:}\ i\,]\,\textbf{acq}\ \mathsf{r}\right.$$
$$\textbf{start}\left\{\ \textbf{for}\ k \in \mathcal{Z}\text{:}\right.$$
$$[\,\mathsf{y}\text{:}\ k\,]\ \textbf{fin}\ [\,\mathsf{x}\text{:}\ i \times i + 1\,]$$
$$\textbf{fin}\ [\,\mathsf{y}\text{:}\ 0\,]\ \textbf{rel}\ \mathsf{r}\ \textbf{halt}\ 0$$

$$\Rightarrow\ \textbf{start}\left\{\ \textbf{for}\ i \in \mathcal{Z}\text{:}\ [\,\mathsf{x}\text{:}\ i\,]\,\textbf{acq}\ \mathsf{r}\right.$$
$$\textbf{start}\left\{\ \textbf{for}\ k \in \mathcal{Z}\text{:}\right.$$
$$[\,\mathsf{y}\text{:}\ k\,]\ \textbf{fin}\ [\,\mathsf{x}\text{:}\ i \times i + 1 \mid \mathsf{y}\text{:}\ 0\,]\ \textbf{rel}\ \mathsf{r}\ \textbf{halt}\ 0$$