Grainless Semantics without Critical Regions

John C. Reynolds
Department of Computer Science
Carnegie Mellon University

April 11, 2007 (corrected April 27, 2007)

(Work in progress, jointly with Ruy Ley-Wild)

(Research partially supported by National Science Foundation Grants CCR-0204242 and CCF-0541021, by the Basic Research in Computer Science Centre of the Danish National Research Foundation, and by EPSRC Visiting Fellowships at Queen Mary, University of London, and Edinburgh University.)

The Problem

What is the meaning of

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

Are the assignments atomic, so that it is either

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

or are evaluation and store operations atomic:

$$(\mathsf{t}_1 := \mathsf{x} \times \mathsf{x}\,;\, \mathsf{x} := \mathsf{t}_1) \parallel (\mathsf{t}_2 := \mathsf{x} + 1\,;\, \mathsf{x} := \mathsf{t}_2)\,?$$

or is each lookup and store atomic:

$$(\mathsf{t}_1 := \mathsf{x}\,;\, \mathsf{t}_2 := \mathsf{x}\,;\, \mathsf{x} := \mathsf{t}_1 \times \mathsf{t}_2) \parallel (\mathsf{t}_3 := \mathsf{x}\,;\, \mathsf{x} := \mathsf{t}_3 + 1)\,?$$

or is the granularity even finer:

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

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

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

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.

Three Principles for Grainless Concurrency

- All operations except locking and unlocking have duration, and can overlap one another during execution.

- If two overlapping operations touch the same location, the meaning of program execution is **wrong**.

- If, from a given starting state, execution of a program can give **wrong**, then no other possibilities need be considered.

Examples

$$
\begin{aligned}
\mathsf{y} := \mathsf{x} - \mathsf{x} \quad &\not\simeq \quad \mathsf{y} := 0 \\
\mathsf{x} := \mathsf{x} + 1 \,;\, \mathsf{x} := \mathsf{x} + 2 \quad &\simeq \quad \mathsf{x} := \mathsf{x} + 3 \\
\mathsf{x} := \mathsf{x} + 1 \,;\, \mathsf{y} := \mathsf{y} + 2 \quad &\simeq \quad \mathsf{y} := \mathsf{y} + 2 \,;\, \mathsf{x} := \mathsf{x} + 1 \\
[\mathsf{x}] := [\mathsf{x}] + 1 \,;\, [\mathsf{y}] := [\mathsf{y}] + 2 \quad &\simeq \quad [\mathsf{y}] := [\mathsf{y}] + 2 \,;\, [\mathsf{x}] := [\mathsf{x}] + 1 \\
\mathsf{x} := 0 \ \mathbf{or}\ \mathsf{y} := 0 \quad &\simeq \quad (\mathsf{x} := 0 \,;\, \mathsf{y} := \mathsf{y}) \\
&\qquad\quad \mathbf{or}\ (\mathsf{y} := 0 \,;\, \mathsf{x} := \mathsf{x})
\end{aligned}
$$

The Programming Language

We begin with the simple imperative language:

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

and add lookup and mutation operations:

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

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

nondeterminism:

$$\langle \text{comm} \rangle ::= \langle \text{comm} \rangle \ \mathbf{or} \ \langle \text{comm} \rangle$$

and concurrent composition:

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

What's Missing?

- Critical Regions

- Allocation and Deallocation

- Passivity

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.

- $\sigma \perp \sigma'$ when $\operatorname{dom} \sigma$ and $\operatorname{dom} \sigma'$ are disjoint.

- $[\,\sigma \mid \ell{:}\, n\,]$ for the state such that

$$
\begin{aligned}
\operatorname{dom}[\,\sigma \mid \ell{:}\, n\,] &= \operatorname{dom} \sigma \cup \{\ell\} \\
[\,\sigma \mid \ell{:}\, n\,](\ell) &= n \\
[\,\sigma \mid \ell{:}\, n\,](\ell') &= \sigma(\ell') \text{ when } \ell \neq \ell'.
\end{aligned}
$$

Semantics of Expressions

$$[\![\langle \mathsf{exp} \rangle]\!] \in \mathsf{States} \to \mathcal{Z} \cup \{\mathbf{wrong}\}$$
$$[\![\langle \mathsf{boolexp} \rangle]\!] \in \mathsf{States} \to \mathsf{Bool} \cup \{\mathbf{wrong}\}$$

$$[\![n]\!]\sigma = n$$
$$[\![v]\!]\sigma = \begin{cases} \sigma\, v & \text{when } v \in \operatorname{dom} \sigma \\ \mathbf{wrong} & \text{otherwise} \end{cases}$$

$$[\![e + e']\!]\sigma = \begin{cases} [\![e]\!]\sigma + [\![e']\!]\sigma & \textsf{when } [\![e]\!]\sigma \neq \mathbf{wrong} \\ & \quad \textsf{and } [\![e']\!]\sigma \neq \mathbf{wrong} \\ \mathbf{wrong} & \textsf{otherwise} \end{cases}$$

$$[\![[e]]\!]\sigma = \begin{cases} \sigma([\![e]\!]\sigma) & \textsf{when } [\![e]\!]\sigma \neq \mathbf{wrong} \\ & \quad \textsf{and } [\![e]\!]\sigma \in \operatorname{\textsf{dom}} \sigma \\ \mathbf{wrong} & \textsf{otherwise} \end{cases}$$

Semantics of Commands

$$[\![\langle\text{comm}\rangle]\!] \in \text{States} \to$$
$$\{S \in \mathcal{P}(\text{States} \cup \{\perp\}) \mid S \neq \{\} \text{ and } S \text{ infinite } \Rightarrow \perp \in S\}$$
$$\cup \{\mathbf{wrong}\}$$

where

$$\begin{aligned} f \sqsubseteq f' \text{ iff } \forall\sigma.\quad & f\,\sigma = f'\,\sigma \\ & \text{or } (\perp \in f\,\sigma \text{ and } f\,\sigma - \{\perp\} \subseteq f'\,\sigma) \\ & \text{or } (\perp \in f\,\sigma \text{ and } f'\,\sigma = \mathbf{wrong}) \end{aligned}$$

Since there is no allocation or deallocation,

$[\![c]\!]\sigma \neq \mathbf{wrong}$ and $\sigma' \in [\![c]\!]\sigma - \{\perp\}$ implies $\text{dom}\,\sigma' = \text{dom}\,\sigma$.

(The domain following $\to$ is formed from the Plotkin powerdomain of the flat domain $(\text{States} \cup \{\mathbf{wrong}\})_\perp$ by the unique retraction that identifies all state sets containing $\mathbf{wrong}$ but no other state sets.)

$$
[\![v := e]\!]\sigma = \begin{cases} \{[\,\sigma \mid v\colon [\![e]\!]\sigma\,]\} & \textsf{when } [\![e]\!]\sigma \neq \mathbf{wrong} \\ & \quad \textsf{and } v \in \textsf{dom}\,\sigma \\ \mathbf{wrong} & \textsf{otherwise} \end{cases}
$$

$$
[\![[e] := e']\!]\sigma = \begin{cases} \{[\,\sigma \mid [\![e]\!]\sigma\colon [\![e']\!]\sigma\,]\} & \textsf{when } [\![e]\!]\sigma \neq \mathbf{wrong} \\ & \quad \textsf{and } [\![e']\!]\sigma \neq \mathbf{wrong} \\ & \quad \textsf{and } [\![e]\!]\sigma \in \textsf{dom}\,\sigma \\ \mathbf{wrong} & \textsf{otherwise} \end{cases}
$$

$$
[\![c \ \mathbf{or}\ c']\!]\sigma = \begin{cases} [\![c]\!]\sigma \cup [\![c']\!]\sigma & \text{when } [\![c]\!]\sigma \neq \mathbf{wrong} \\ & \quad\text{and } [\![c']\!]\sigma \neq \mathbf{wrong} \\ \mathbf{wrong} & \text{otherwise} \end{cases}
$$

$$
[\![c\, ;\, c']\!]\sigma = \begin{cases} \mathbf{wrong} & \text{when } [\![c]\!]\sigma = \mathbf{wrong} \\ & \text{or } \exists\sigma' \in [\![c]\!]\sigma - \{\bot\}.\ [\![c']\!]\sigma' = \mathbf{wrong} \\ \bigcup\{\, \mathbf{if}\ \widehat{\sigma}' = \bot\ \mathbf{then}\ \bot\ \mathbf{else}\ [\![c']\!]\widehat{\sigma}' \mid \widehat{\sigma}' \in [\![c]\!]\sigma \,\} & \\ & \text{otherwise} \end{cases}
$$

Concurrent Composition

If, for all σ_0 and σ_1 such that $\sigma = \sigma_0 \cup \sigma_1$ and $\sigma_0 \perp \sigma_1$, either $[\![c_0]\!]\sigma_0 = \mathbf{wrong}$ or $[\![c_1]\!]\sigma_1 = \mathbf{wrong}$, then:

$$[\![c_0 \parallel c_1]\!]\sigma = \mathbf{wrong}.$$

Otherwise:

$$[\![c_0 \parallel c_1]\!]\sigma =$$

$$\left\{ \begin{array}{ll|l} \perp & \hat{\sigma}'_0 = \perp & \sigma = \sigma_0 \cup \sigma_1 \text{ and } \sigma_0 \perp \sigma_1 \\ & \quad \text{or } \hat{\sigma}'_1 = \perp & \text{and } [\![c_0]\!]\sigma_0 \neq \mathbf{wrong} \\ & & \text{and } [\![c_1]\!]\sigma_1 \neq \mathbf{wrong} \\ \hat{\sigma}'_0 \cup \hat{\sigma}'_1 & \text{otherwise} & \text{and } \hat{\sigma}'_0 \in [\![c_0]\!]\sigma_0 \text{ and } \hat{\sigma}'_1 \in [\![c_1]\!]\sigma_1 \end{array} \right\}$$

Safety Monotonicity

If $\sigma \subseteq \sigma'$ and $[\![c]\!]\sigma \neq \mathbf{wrong}$, then $[\![c]\!]\sigma' \neq \mathbf{wrong}$.

Strong Frame Property

$\sigma \subseteq \sigma'$ and $[\![c]\!]\sigma \neq \mathbf{wrong}$ implies

$$[\![c]\!]\sigma' = \{\, \mathbf{if}\ \hat{\sigma} = \bot\ \mathbf{then}\ \bot\ \mathbf{else}\ \hat{\sigma} \cup (\sigma' - \sigma) \mid \hat{\sigma} \in [\![c]\!]\sigma \,\}.$$

(This is stronger than O'Hearn's frame property since we are not considering allocation.)

Footprints

We define $\mathcal{F}(c)$ to be the set of *footprints* of c, which are the minimal starting states for which the execution of c does not go wrong, i.e., $\sigma_f \in \mathcal{F}(c)$ iff:

- $[\![c]\!]\sigma_f \neq \mathbf{wrong}$, and
- for all proper $\sigma \subset \sigma_f$, $[\![c]\!]\sigma = \mathbf{wrong}$.

Footprints of Expressions

$$
\begin{aligned}
\mathcal{F}(n) &= \{[\,]\} \\
\mathcal{F}(v) &= \{\,[\,v\colon n\,] \mid n \in \mathcal{Z}\,\} \\
\mathcal{F}(e + e') &= \{\,\sigma \cup \sigma' \mid \\
&\qquad\qquad \sigma \in \mathcal{F}(e) \text{ and } \sigma' \in \mathcal{F}(e') \text{ and } \sigma \smile \sigma'\,\} \\
\mathcal{F}([e]) &= \{\,\sigma \cup [\,[\![e]\!]\sigma\colon n\,] \mid \\
&\qquad\qquad \sigma \in \mathcal{F}(e) \text{ and } n \in \mathcal{Z} \text{ and } \sigma \smile [\,[\![e]\!]\sigma\colon n\,]\,\}
\end{aligned}
$$

Footprints of Commands

$$\begin{aligned}
\mathcal{F}(v := e) &= \{\, \sigma \cup [\, v\colon n \,] \mid \\
&\qquad \sigma \in \mathcal{F}(e) \text{ and } n \in \mathcal{Z} \text{ and } \sigma \smile [\, v\colon n \,] \,\} \\
\mathcal{F}([e] := e') &= \{\, \sigma \cup \sigma' \cup [\, [\![e]\!]\sigma\colon n \,] \mid \\
&\qquad \sigma \in \mathcal{F}(e) \text{ and } \sigma' \in \mathcal{F}(e') \text{ and } n \in \mathcal{Z} \\
&\qquad \text{and } \sigma \smile \sigma' \text{ and } \sigma \cup \sigma' \smile [\, [\![e]\!]\sigma\colon n \,] \,\} \\
\mathcal{F}(c \;\mathbf{or}\; c') &= \{\, \sigma \cup \sigma' \mid \\
&\qquad \sigma \in \mathcal{F}(c) \text{ and } \sigma' \in \mathcal{F}(c') \text{ and } \sigma \smile \sigma' \,\} \\
\mathcal{F}(c \parallel c') &= \{\, \sigma \cup \sigma' \mid \\
&\qquad \sigma \in \mathcal{F}(c) \text{ and } \sigma' \in \mathcal{F}(c') \text{ and } \sigma \perp \sigma' \,\}
\end{aligned}$$

Sequential Composition

$$
\begin{aligned}
\mathcal{F}(c\,;c') = \{\, \sigma \cup \textstyle\bigcup_{i=1}^{n}(\sigma'_i - \sigma_i) \mid \\
&\sigma \in \mathcal{F}(c) \text{ and } \{\sigma_1, \ldots, \sigma_n\} = [\![c]\!]\sigma \text{ and} \\
&\forall i \in 1 \text{ to } n.\, (\sigma'_i \in \mathcal{F}(c') \text{ and } \sigma'_i \smile \sigma_i) \text{ and} \\
&\forall i, j \in 1 \text{ to } n.\, (\sigma'_i - \sigma_i) \smile (\sigma'_j - \sigma_j)\,\}
\end{aligned}
$$

Properties of Footprints

We write σ_f, σ'_f for footprints of c. Then

- $\sigma_f \subseteq \sigma$ implies $[\![c]\!]\sigma \neq \mathbf{wrong}$.

- $(\forall \sigma_f \in \mathcal{F}(c).\ \sigma_f \not\subseteq \sigma)$ implies $[\![c]\!]\sigma = \mathbf{wrong}$.

- $\sigma_f \smile \sigma$ and $\sigma_f \not\subseteq \sigma$ implies $[\![c]\!]\sigma = \mathbf{wrong}$.
- $\sigma_f \smile \sigma'_f$ implies $\sigma_f = \sigma'_f$.
- $\sigma_f \subseteq \sigma$ and $\sigma'_f \subseteq \sigma$ implies $\sigma_f = \sigma'_f$.

In Summary

If $\forall \sigma_f \in \mathcal{F}(c).\ \sigma_f \not\subseteq \sigma$, then

$$[\![c]\!]\sigma = \mathbf{wrong}.$$

Otherwise, there is a unique $\sigma_f \in \mathcal{F}(c)$ such that $\sigma_f \subseteq \sigma$, and

$$[\![c]\!]\sigma = \{\, \mathbf{if}\ \widehat{\sigma} = \bot\ \mathbf{then}\ \bot\ \mathbf{else}\ \widehat{\sigma} \cup (\sigma - \sigma_f) \mid \widehat{\sigma} \in [\![c]\!]\sigma_f \,\}.$$

Concurrent Composition is Determinate

Recall that, if there are any σ_0 and σ_1 such that $\sigma = \sigma_0 \cup \sigma_1$, $\sigma_0 \perp \sigma_1$, $[\![c_0]\!]\sigma_0 \neq \mathbf{wrong}$, and $[\![c_1]\!]\sigma_1 \neq \mathbf{wrong}$, then:

$$[\![c_0 \parallel c_1]\!]\sigma = \left\{ \begin{cases} \perp & \begin{array}{l} \widehat{\sigma}'_0 = \perp \\ \quad \text{or } \widehat{\sigma}'_1 = \perp \end{array} \\ \widehat{\sigma}'_0 \cup \widehat{\sigma}'_1 & \text{otherwise} \end{cases} \;\middle|\; \begin{array}{l} \sigma = \sigma_0 \cup \sigma_1 \text{ and } \sigma_0 \perp \sigma_1 \\ \text{and } [\![c_0]\!]\sigma_0 \neq \mathbf{wrong} \\ \text{and } [\![c_1]\!]\sigma_1 \neq \mathbf{wrong} \\ \text{and } \widehat{\sigma}'_0 \in [\![c_0]\!]\sigma_0 \text{ and } \widehat{\sigma}'_1 \in [\![c_1]\!]\sigma_1 \end{array} \right\}$$

In fact, if

$$\begin{array}{lcl} \sigma = \sigma_0 \cup \sigma_1 \text{ and } \sigma_0 \perp \sigma_1 & & \sigma = \sigma_0' \cup \sigma_1' \text{ and } \sigma_0' \perp \sigma_1' \\ \text{and } [\![c_0]\!]\sigma_0 \neq \mathbf{wrong} & \text{and} & \text{and } [\![c_0]\!]\sigma_0' \neq \mathbf{wrong} \\ \text{and } [\![c_1]\!]\sigma_1 \neq \mathbf{wrong} & & \text{and } [\![c_1]\!]\sigma_1' \neq \mathbf{wrong} \end{array}$$

then

$$\left\{ \begin{array}{ll} \perp & \hat{\sigma}_0' = \perp \text{ or } \hat{\sigma}_1' = \perp \\ \hat{\sigma}_0' \cup \hat{\sigma}_1' & \text{otherwise} \end{array} \;\middle|\; \begin{array}{l} \hat{\sigma}_0' \in [\![c_0]\!]\sigma_0 \\ \quad \text{and } \hat{\sigma}_1' \in [\![c_1]\!]\sigma_1 \end{array} \right\} =$$

$$\left\{ \begin{array}{ll} \perp & \hat{\sigma}_0' = \perp \text{ or } \hat{\sigma}_1' = \perp \\ \hat{\sigma}_0' \cup \hat{\sigma}_1' & \text{otherwise} \end{array} \;\middle|\; \begin{array}{l} \hat{\sigma}_0' \in [\![c_0]\!]\sigma_0' \\ \quad \text{and } \hat{\sigma}_1' \in [\![c_1]\!]\sigma_1' \end{array} \right\}$$

If $[\![c_0]\!]\sigma_0$ and $[\![c_1]\!]\sigma_1$ are singletons, then so is $[\![c_0 \| c_1]\!]\sigma$.