15–819A3 — Class Notes 3
Reasoning about Low-Level Programming Languages

John C. Reynolds
Department of Computer Science
Carnegie Mellon University

February 15, 2001

Intuitionistic Assertions

We say that an assertion p is *intuitionistic* iff, for all stores s and heaps h and h',

$$h \subseteq h' \text{ and } [\![p]\!]_{\text{assert}}(s, h) \text{ implies } [\![p]\!]_{\text{assert}}(s, h').$$

An assertion p is intuitionistic iff

$$p * \mathbf{true} \Rightarrow p$$

is valid. Note that the opposite implication is always valid, since

$$\begin{aligned} p &\Leftrightarrow p * \mathbf{emptyheap} \\ &\Rightarrow p * \mathbf{true}. \end{aligned}$$

We define:

$$\begin{aligned} \overset{\mathrm{i}}{\neg}\, p &\overset{\text{def}}{=} \mathbf{true} \mathrel{-\!\!*} (\neg\, p) \\ p \overset{\mathrm{i}}{\Rightarrow} q &\overset{\text{def}}{=} \mathbf{true} \mathrel{-\!\!*} (p \Rightarrow q) \\ p \overset{\mathrm{i}}{\Leftrightarrow} q &\overset{\text{def}}{=} \mathbf{true} \mathrel{-\!\!*} (p \Leftrightarrow q) \end{aligned}$$

Intuitionistic Assertions (continued)

If p and q are intuitionistic assertions, e and e' are expressions, and r is any assertion, then

$$\begin{gathered}
\text{Any pure assertion} \\
e \hookrightarrow e' \\
\overset{\mathrm{i}}{\neg}\, p \\
p \wedge q \\
p \vee q \\
p \overset{\mathrm{i}}{\Rightarrow} q \\
p \overset{\mathrm{i}}{\Leftrightarrow} q \\
\forall v.\ p \\
\exists v.\ p \\
p \ * \ q \\
p \mathrel{-\!\!*} q \\
r \ * \ \mathbf{true}.
\end{gathered}$$

Notice what is missing: $\mapsto$, **emptyheap**, $\neg$, $\Rightarrow$, and $\Leftrightarrow$.

Also notice that $\overset{\mathrm{i}}{\neg}$ does not obey the law of the excluded middle. For example,

$$\mathsf{x} \hookrightarrow 13 \qquad\qquad \overset{\mathrm{i}}{\neg}\,(\mathsf{x} \hookrightarrow 13)$$

are both false when the value of x is not in the domain of the heap.

Trees (Binary)

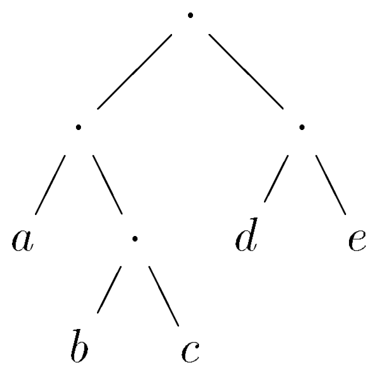
$\tau \in$ Trees iff

$$\tau \in \text{Atoms}$$

$$\text{or } \tau = (\tau_1, \tau_2) \text{ where } \tau_1, \tau_2 \in \text{Trees}$$

e.g.

$$((a,(b,c)),(d,e)) =$$



Tree Representation (without sharing)

For $\tau \in$ Trees, we define the assertion

$$\mathsf{tree}\,\tau\,(i)$$

by structural induction:

$$\mathsf{tree}\,a\,(i) \text{ iff } \mathbf{emptyheap} \wedge i = a$$

$$\begin{aligned}&\mathsf{tree}\,(\tau_1, \tau_2)\,(i) \text{ iff}\\ &\quad \exists i_1, i_2.\;\; i \mapsto i_1, i_2 \;*\; \mathsf{tree}\,\tau_1\,(i_1) \;*\; \mathsf{tree}\,\tau_2\,(i_2)\end{aligned}$$

Tree Representation (with sharing)

For $\tau \in$ Trees, we define

$$\mathsf{stree}\,\tau\,(i)$$

by:

$$\mathsf{stree}\,a\,(i) \text{ iff } i = a$$

$$\begin{aligned}&\mathsf{stree}\,(\tau_1, \tau_2)\,(i) \text{ iff}\\ &\quad \exists i_1, i_2.\;\; i \mapsto i_1, i_2 \;*\; (\mathsf{stree}\,\tau_1\,(i_1) \wedge \mathsf{stree}\,\tau_2\,(i_2))\end{aligned}$$

Notice that $\mathsf{stree}\,\tau\,(i)$ is intuitionistic.

The proof is by induction on the structure of τ. If τ is an atom a,

$$\begin{aligned}
&\mathsf{stree}\,a\,(i) \;*\; \mathbf{true} \\
&\qquad \Rightarrow i = a \;*\; \mathbf{true} \\
&\qquad \Rightarrow i = a \\
&\qquad \Rightarrow \mathsf{stree}\,a\,(i).
\end{aligned}$$

since $i = a$ is pure. Otherwise, $\tau = (\tau_1, \tau_2)$, and

$$\begin{aligned}
&\mathsf{stree}\,(\tau_1,\tau_2)\,(i) \;*\; \mathbf{true} \\
&\quad \Rightarrow \exists i_1, i_2.\;\; i \mapsto i_1, i_2 \;*\; (\mathsf{stree}\,\tau_1\,(i_1) \wedge \mathsf{stree}\,\tau_2\,(i_2)) \;*\; \mathbf{true} \\
&\quad \Rightarrow \exists i_1, i_2.\;\; i \mapsto i_1, i_2 \;*\; \\
&\qquad\qquad ((\mathsf{stree}\,\tau_1\,(i_1) \;*\; \mathbf{true}) \wedge (\mathsf{stree}\,\tau_2\,(i_2) \;*\; \mathbf{true})) \\
&\quad \Rightarrow \exists i_1, i_2.\;\; i \mapsto i_1, i_2 \;*\; (\mathsf{stree}\,\tau_1\,(i_1) \wedge \mathsf{stree}\,\tau_2\,(i_2)) \\
&\quad \Rightarrow \mathsf{stree}\,(\tau_1,\tau_2)\,(i),
\end{aligned}$$

by the induction hypothesis for τ_1 and τ_2.

A Problem

Suppose we wish to prove that a recursively defined procedure $\mathsf{copytree}$ satisfies

$$\{\mathsf{stree}\,\tau(\mathsf{i})\}$$
$$\mathsf{copytree}(\mathsf{i};\mathsf{j})$$
$$\{\mathsf{stree}\,\tau(\mathsf{i}) \ast \mathsf{tree}\,\tau(\mathsf{j})\}$$

Then, we must use this specification as a hypothesis in proving that the recursive calls in the procedure body satisfy:

$$\{\mathsf{stree}\,\tau_1(\mathsf{i}_1) \wedge \mathsf{stree}\,\tau_2(\mathsf{i}_2)\}$$
$$\mathsf{copytree}(\mathsf{i}_1;\mathsf{j}_1)$$
$$\{(\mathsf{stree}\,\tau_1(\mathsf{i}_1) \wedge \mathsf{stree}\,\tau_2(\mathsf{i}_2)) \ast \mathsf{tree}\,\tau_1(\mathsf{j}_1)\}$$
$$\mathsf{copytree}(\mathsf{i}_2;\mathsf{j}_2)$$
$$\{(\mathsf{stree}\,\tau_1(\mathsf{i}_1) \wedge \mathsf{stree}\,\tau_2(\mathsf{i}_2)) \ast \mathsf{tree}\,\tau_1(\mathsf{j}_1) \ast \mathsf{tree}\,\tau_2(\mathsf{j}_2)\}$$

But the hypothesis is not strong enough to imply this.

Possible solutions:

- Introduce passivity.
- Introduce auxiliary variables denoting assertions:

$$\{\mathsf{p} \wedge \mathsf{stree}\,\tau(\mathsf{i})\}$$
$$\mathsf{copytree}(\mathsf{i};\mathsf{j})$$
$$\{\mathsf{p} \ast \mathsf{tree}\,\tau(\mathsf{j})\}$$

- Introduce auxiliary variables denoting heaps:

$$\{\mathbf{this}(\mathsf{h}) \wedge \mathsf{stree}\,\tau(\mathsf{i})\}$$
$$\mathsf{copytree}(\mathsf{i};\mathsf{j})$$
$$\{\mathbf{this}(\mathsf{h}) \ast \mathsf{tree}\,\tau(\mathsf{j})\}$$

A Tree-Copying Procedure

The procedure

$$
\begin{array}{l}
\mathsf{copytree}(\mathsf{i};\mathsf{j}) = \\
\quad \textbf{if } \mathsf{isatom}(\mathsf{i}) \textbf{ then } \mathsf{j} := \mathsf{i} \textbf{ else} \\
\quad\quad \textbf{newvar } \mathsf{i}_1, \mathsf{i}_2, \mathsf{j}_1, \mathsf{j}_2 \textbf{ in} \\
\quad\quad\quad \big(\mathsf{i}_1 := [\mathsf{i}] \; ; \mathsf{i}_2 := [\mathsf{i} + 1] \; ; \\
\quad\quad\quad \textbf{copytree}(\mathsf{i}_1;\mathsf{j}_1) \; ; \textbf{copytree}(\mathsf{i}_2;\mathsf{j}_2) \; ; \mathsf{j} := \textbf{cons}(\mathsf{j}_1, \mathsf{j}_2)\big)
\end{array}
$$

satisfies

$$
\begin{array}{l}
\{\mathsf{p} \wedge \mathsf{stree}\,\tau(\mathsf{i})\} \\
\mathsf{copytree}(\mathsf{i};\mathsf{j}) \\
\{\mathsf{p} \, * \, \mathsf{tree}\,\tau(\mathsf{j})\}
\end{array}
$$

since

$$
\begin{array}{l}
\{\mathsf{p} \wedge \mathsf{stree}\,\tau(\mathsf{i})\} \\
\textbf{if } \mathsf{isatom}(\mathsf{i}) \textbf{ then} \\
\quad \{\mathsf{p} \wedge \mathsf{isatom}(\tau) \wedge \tau = \mathsf{i}\} \\
\quad \{\mathsf{p} \ast (\mathsf{isatom}(\tau) \wedge \tau = \mathsf{i} \wedge \textbf{emptyheap})\} \\
\quad \mathsf{j} := \mathsf{i} \\
\quad \{\mathsf{p} \ast (\mathsf{isatom}(\tau) \wedge \tau = \mathsf{j} \wedge \textbf{emptyheap})\} \\
\textbf{else} \\
\quad \{\exists \tau_1, \tau_2.\ \tau = (\tau_1, \tau_2) \wedge \mathsf{p} \wedge \mathsf{stree}\,(\tau_1, \tau_2)(\mathsf{i})\} \\
\quad \textbf{newvar } \mathsf{i}_1, \mathsf{i}_2, \mathsf{j}_1, \mathsf{j}_2 \textbf{ in} \\
\quad\quad \big(\mathsf{i}_1 := [\mathsf{i}]\ ;\ \mathsf{i}_2 := [\mathsf{i}+1]\ ; \\
\quad\quad \{\exists \tau_1, \tau_2.\ \tau = (\tau_1, \tau_2) \wedge \\
\quad\quad\quad \mathsf{p} \wedge (\mathsf{i} \mapsto \mathsf{i}_1, \mathsf{i}_2 \ast (\mathsf{stree}\,\tau_1\,(\mathsf{i}_1) \wedge \mathsf{stree}\,\tau_2\,(\mathsf{i}_2)))\} \\
\quad\quad \{\exists \tau_1, \tau_2.\ \tau = (\tau_1, \tau_2) \wedge \mathsf{p} \wedge \mathsf{stree}\,\tau_1\,(\mathsf{i}_1) \wedge \mathsf{stree}\,\tau_2\,(\mathsf{i}_2)\} \\
\quad\quad \textbf{copytree}(\mathsf{i}_1; \mathsf{j}_1)\ ; \\
\quad\quad \{\exists \tau_1, \tau_2.\ (\tau = (\tau_1, \tau_2) \wedge \mathsf{p} \wedge \mathsf{stree}\,\tau_2\,(\mathsf{i}_2)) \ast \mathsf{tree}\,\tau_1\,(\mathsf{j}_1)\} \\
\quad\quad \textbf{copytree}(\mathsf{i}_2; \mathsf{j}_2)\ ; \\
\quad\quad \{\exists \tau_1, \tau_2.\ (\tau = (\tau_1, \tau_2) \wedge \mathsf{p}) \ast \mathsf{tree}\,\tau_1\,(\mathsf{j}_1) \ast \mathsf{tree}\,\tau_2\,(\mathsf{j}_2)\} \\
\quad\quad \mathsf{j} := \textbf{cons}(\mathsf{j}_1, \mathsf{j}_2) \\
\quad\quad \{\exists \tau_1, \tau_2.\ (\tau = (\tau_1, \tau_2) \wedge \mathsf{p}) \ast \\
\quad\quad\quad \mathsf{j} \mapsto \mathsf{j}_1, \mathsf{j}_2 \ast \mathsf{tree}\,\tau_1\,(\mathsf{j}_1) \ast \mathsf{tree}\,\tau_2\,(\mathsf{j}_2)\} \\
\quad\quad \{\exists \tau_1, \tau_2.\ (\tau = (\tau_1, \tau_2) \wedge \mathsf{p}) \ast \mathsf{tree}\,(\tau_1, \tau_2)\,(\mathsf{j})\}\big) \\
\{(\mathsf{p} \ast \mathsf{tree}\,\tau(\mathsf{j})\}
\end{array}
$$

The General Case

Suppose

$$\{P \ast Q\}\ C\ \{P \ast R\} \tag{1}$$

holds, and C does not mutate or deallocate the part of the heap described by P. Then

$$\{(v \wedge P) \ast Q\}\ C\ \{v \ast R\} \tag{2}$$

holds (where v is an assertion variable), and even the stronger

$$\{(v \wedge (P \ast \mathbf{true})) \ast Q\}\ C\ \{v \ast R\} \tag{3}$$

holds.

Since $P \Rightarrow P \ast \mathbf{true}$ is valid, one can obtain (2) from (3). Simliarly, if P is intuitionistic, so that $P \ast \mathbf{true} \Rightarrow P$ is valid, then one can obtain (3) from (2).

One can obtain (1) from (2) by substituting P for v.

Tree Patterns

$\tau \in \text{Trees}(n)$ iff

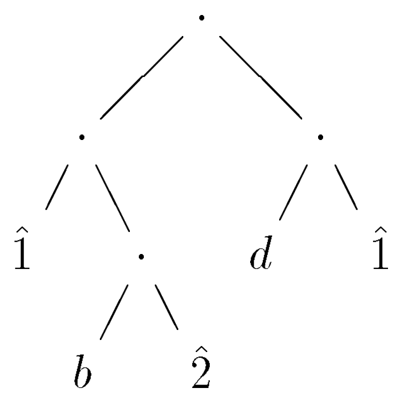
$$\tau = \hat{k} \text{ where } 1 \leq k \leq n$$

$$\tau \in \text{Atoms}$$

$$\text{or } \tau = (\tau_1, \tau_2) \text{ where } \tau_1, \tau_2 \in \text{Trees}(n)$$

e.g.

$$((\hat{1}, (b, \hat{2})), (d, \hat{1})) =$$



Tree Substitution

Suppose $\tau \in \text{Trees}(m)$ and σ is a sequence of m trees in $\text{Trees}(n)$. Then we define $\tau/\sigma \in \text{Trees}(n)$ by:

$$\hat{k}/\sigma = \sigma_k$$

$$\mathsf{a}/\sigma = \mathsf{a}$$

$$(\tau_1, \tau_2)/\sigma = (\tau_1/\sigma, \tau_2/\sigma)$$

For example,

$$((\hat{1}, (b, \hat{2})), (d, \hat{1}))/\langle a, (f, \hat{1})\rangle = ((a, (b, (f, \hat{1}))), (d, a))$$

Representation of Tree Patterns (without sharing)

Suppose α is a sequence of numbers.

For $\tau \in \text{Trees}(\#\alpha)$, we define the assertion

$$\mathsf{tree}\,\tau\,(i : \alpha)$$

by structural induction:

$$\mathsf{tree}\,\hat{k}\,(i : \alpha) \text{ iff } \mathbf{emptyheap} \wedge i = \alpha_k$$

$$\mathsf{tree}\,a\,(i : \alpha) \text{ iff } \mathbf{emptyheap} \wedge i = a$$

$$\begin{aligned}\mathsf{tree}\,(\tau_1, \tau_2)\,(i : \alpha) \text{ iff } & \exists i_1, i_2.\ \ i \mapsto i_1, i_2 \\ & * \ \mathsf{tree}\,\tau_1\,(i_1 : \alpha) \ * \ \mathsf{tree}\,\tau_2\,(i_2 : \alpha)\end{aligned}$$

Representation with Sharing

For $\tau \in \text{Trees}(\#\alpha)$, we define

$$\mathsf{stree}\,\tau\,(i : \alpha)$$

by:

$$\mathsf{stree}\,\hat{k}\,(i : \alpha) \text{ iff } i = \alpha_k$$

$$\mathsf{stree}\,a\,(i : \alpha) \text{ iff } i = a$$

$$\begin{aligned}\mathsf{stree}\,(\tau_1, \tau_2)\,(i : \alpha) \text{ iff } & \exists i_1, i_2.\ \ i \mapsto i_1, i_2 \\ & * \ \big(\mathsf{stree}\,\tau_1\,(i_1 : \alpha) \wedge \mathsf{stree}\,\tau_2\,(i_2 : \alpha)\big)\end{aligned}$$