Bisimilarity as a Theory of Functional Programming. Mini-Course.
Andrew D. Gordon
BRICS Note NS–95–3

Errata found by November 24, 1995

Page 6 The structural rule (Red Experiment) is missing! In Definition 2.5, small-step reduction should be defined by:

> The **small-step reduction** relation $a \rightsquigarrow b$ is given inductively by the (Red $-$) rules of the language together with the generic rule
>
> $$\frac{a \rightsquigarrow b}{\mathcal{E}[a] \rightsquigarrow \mathcal{E}[b]} \text{ (Red Experiment)}$$

Page 11 The proof of Proposition 2.10 is garbled. It should read:

> Make the following definitions.
>
> $$\begin{aligned} \nu_1 &\stackrel{\text{def}}{=} \nu X.\, F(X) \cup \nu \\ \nu_2 &\stackrel{\text{def}}{=} \nu X.\, F(X \cup \nu) \\ \nu_3 &\stackrel{\text{def}}{=} \nu X.\, F(X \cup \nu) \cup \nu \end{aligned}$$
>
> We must show that each ν_i equals ν. Since $\nu \subseteq F(\nu)$ it follows by co-induction that $\nu \subseteq \nu_i$ for each i. The reverse inclusions take a little more work.
>
> **$\nu_2 \subseteq \nu$.** Since ν is a fixpoint of F, which is monotone, we have $\nu = F(\nu) \subseteq F(\nu_2 \cup \nu)$. Now $\nu_2 \subseteq F(\nu_2 \cup \nu)$ so $\nu_2 \cup \nu \subseteq F(\nu_2 \cup \nu)$, and therefore $\nu_2 \cup \nu \subseteq \nu$ by co-induction. Hence $\nu_2 \subseteq \nu$.
>
> **$\nu_1 \subseteq \nu_2$.** We have $\nu_1 = F(\nu_1) \cup \nu = F(\nu_1) \cup F(\nu) \subseteq F(\nu_1 \cup \nu)$. So $\nu_1 \subseteq \nu X.\, F(X \cup \nu)$.
>
> **$\nu_3 \subseteq \nu_2$.** We have $\nu_3 = F(\nu_3 \cup \nu) \cup \nu = F(\nu_3 \cup \nu) \cup F(\nu) = F(\nu_3 \cup \nu)$ since $F(\nu) \subseteq F(\nu_3 \cup \nu)$. Hence $\nu_3 \subseteq \nu X.\, F(X \cup \nu)$.

Page 21 As a clarification insert the following after the proof.

> The converses of (Fun Ext), (Inj Lit), (Inj Fun) and (Inj Stm) hold, and are instances of the congruence property of $\sim$ that we shall prove in the next section (in fact, these particular instances can be proved directly).

Page 24 Lemma 3.3 is incorrect. In a typed setting we need to be careful about the types of variables bound in the scope of the hole in a context. A corrected version of the lemma is given on the next page. For the sake of simplicity I have assumed there is exactly one hole in a variable-capturing context. Hence it makes sense to talk about the environment of bound variables in whose scope lies the hole. Nothing in the rest of the notes depends on the exact statement of Lemma 3.3.

Page 26 Names (Red Appl) and (Red Fun) should both be (Red Beta). The reference to (Red Sound) should be to Proposition 2.18(2).

Page 28 In proof of Theorem 3.8, the reference to Lemma 3.6 should be to Lemma 3.7.

Page 32 In the proof of Proposition 3.19, $b'Cnv$ should be $b'\Downarrow$. After the definition of ν to be $\nu\mathcal{S}.\,F(\approx\mathcal{S}\approx)$ the first occurrence of $\approx\mathcal{S}\approx$ should be $\approx\approx\approx$, and thereafter should be $\approx\nu\approx$.

Page 35 In the first line of the proof of Lemma 4.2, $\mathcal{S}$ should be $\overline{\mathcal{S}}$. In the case (Value Arith), the only transition of a should be $a \xrightarrow{\ell} \mathbf{0}$. In case (Value Arith), $\lambda(x)e'$ should be $\lambda(x{:}A)e'$.

Page 36 The first line of the proof of Proposition 4.4 should read

Since ν_V is $\langle - \rangle_V$ dense, we have $\sim\nu_V\sim \subseteq \sim$ by Lemma 4.3.

Thanks to Søren B. Lassen for pointing out many of these mistakes.

Correction to Lemma 3.3

The following lemma shows that this definition is equivalent to the more common one based on variable-capturing contexts. Let $\mathcal{C}$ stand for a variable-capturing context, that is, an expression, not identified up to alpha-conversion, whose bound variables are distinct, containing a single typed hole denoted by '$-_A$' and subject to the extra type assignment rule $\Gamma \vdash -_A : A$. We write $\mathcal{C}[e]$ for the outcome of filling the hole in the context $\mathcal{C}$ with the expression e. Free variables of e may be bound by this process.

Lemma 3.3 *Assume that $\mathcal{R} \subseteq Rel^\circ$ is a preorder. $\mathcal{R}$ is a precongruence iff* (Cong $\mathcal{R}$) *holds.* (Cong $\mathcal{R}$) *is given by*

$$\frac{\Gamma \vdash \mathcal{C}[-_A] : B \qquad \Gamma, \Gamma' \vdash e \mathrel{\mathcal{R}} e' : A}{\Gamma \vdash \mathcal{C}[e] \mathrel{\mathcal{R}} \mathcal{C}[e'] : B} \text{(Cong } \mathcal{R}\text{)}$$

where Γ' is the list of bound variables in $\mathcal{C}$ whose scope includes the hole.

Proof

(Only If) Suppose that preorder $\mathcal{R}$ satisfies $\widehat{\mathcal{R}} \subseteq \mathcal{R}$. We prove by induction on the derivation of $\Gamma \vdash \mathcal{C}[-_A] : B$ that if Γ' is the environment bound in $\mathcal{C}$, then $\Gamma, \Gamma' \vdash e \mathrel{\mathcal{R}} e'$ implies $\Gamma \vdash \mathcal{C}[e] \mathrel{\mathcal{R}} \mathcal{C}[e']$.

$\mathcal{C}$ is $-_A$. Such a context binds no variables, so $\Gamma' = \varnothing$. We have $\Gamma \vdash e \mathrel{\mathcal{R}} e'$ by assumption.

$\mathcal{C}$ is a variable Impossible because $\mathcal{C}$ has to contain a hole.

$\mathcal{C}$ is an application. Either $\mathcal{C}$ is $\mathcal{C}_0\, e_0$ or $e_0\, \mathcal{C}_0$, where $\mathcal{C}_0$ is a smaller context and e_0 is an ordinary expression (with no hole). In the first case we have $\Gamma \vdash \mathcal{C}_0 : C \to B$ and $\Gamma \vdash e_0 : C$ for some type C. By induction hypothesis we have $\Gamma \vdash \mathcal{C}_0[e] \mathrel{\mathcal{R}} \mathcal{C}_0[e'] : C \to B$ since $\mathcal{C}_0$ binds the same variables as $\mathcal{C}$. By (Comp Appl) and reflexivity we have $\Gamma \vdash \mathcal{C}_0[e]\, e_0 \mathrel{\mathcal{R}} \mathcal{C}_0[e']\, e_0 : B$, that is, $\Gamma \vdash \mathcal{C}[e_0] \mathrel{\mathcal{R}} \mathcal{C}[e'] : B$. The second case, when the context is $e_0\, \mathcal{C}_0$, is similar.

$\mathcal{C}$ is $\lambda(x{:}C_1)\mathcal{C}'$. The environment Γ' of variables bound by $\mathcal{C}$ must be $x{:}C_1, \Gamma''$ with $\Gamma, x{:}C_1 \vdash \mathcal{C}' : C_2$ where $B \equiv C_1 \to C_2$. By induction hypothesis we have $\Gamma, x{:}C_1 \vdash \mathcal{C}'[e] \mathrel{\mathcal{R}} \mathcal{C}'[e'] : C_2$ since $\Gamma, x{:}C_1, \Gamma'' \vdash e \mathrel{\mathcal{R}} e'$ and Γ'' is the variables bound in $\mathcal{C}'$. Rule (Comp Fun) now yields $\Gamma \vdash \mathcal{C}[e] \mathrel{\mathcal{R}} \mathcal{C}[e'] : B$.

Cases for the other forms of contexts follow similarly.

(If) Suppose that preorder $\mathcal{R}$ satisfies (Cong $\mathcal{R}$). We must show that $\widehat{\mathcal{R}} \subseteq \mathcal{R}$. We need to consider all the rules of compatible refinement from Table 7.

(Comp x) $\mathcal{R}$ is reflexive.

(Comp Lit) $\mathcal{R}$ is a reflexive.

(Comp Appl) Suppose that $\Gamma \vdash e_1 \mathcal{R}\, e_1'$ and $\Gamma \vdash e_2 \mathcal{R}\, e_2'$. By (Cong $\mathcal{R}$) we have $\Gamma \vdash e_1(e_2) \mathcal{R}\, e_1'(e_2)$ and $\Gamma \vdash e_1'(e_2) \mathcal{R}\, e_1'(e_2')$ and therefore by transitivity that $\Gamma \vdash e_1(e_2) \mathcal{R}\, e_1'(e_2')$.

(Comp Fun) Suppose that $\Gamma, x{:}A \vdash e \mathcal{R}\, e'$. Using context $\lambda(x{:}A)-$ we have $\Gamma \vdash \lambda(x{:}A)e\, \widehat{\mathcal{R}}\, \lambda(x{:}A)e'$ from (Cong $\mathcal{R}$).

The other cases are similar.

■