Polishing Up the Church–Rosser Theorem

Randy Pollack

Version of October 26, 2011

Confluence

- Let $R \in \mathit{Rel2}$, the class of binary relations.
- the **transitive-reflexive closure** of R, written R^*, is defined inductively by

$$\frac{aRb}{aR^*b}\ *\text{-}\mathit{base} \qquad \frac{}{aR^*a}\ *\text{-}\mathit{refl} \qquad \frac{aR^*b \quad bR^*c}{aR^*c}\ *\text{-}\mathit{trans}.$$

- R has the **diamond property**, $dp(R)$, iff

$$\forall a, b, c \,.\, aRb \ \wedge\ aRc \ \Rightarrow\ \exists d \,.\, bRd \ \wedge\ cRd.$$

$$\begin{tikzcd} & a \arrow[dl, "R"'] \arrow[dr, "R"] & \\ b \arrow[dr, dashed, "R"'] & & c \arrow[dl, dashed, "R"] \\ & d & \end{tikzcd}$$

- R is **confluent** iff $dp(R^*)$.

Recall some λ-calculus

- $x, y, z, \ldots,$ range over variables.
- λ-terms are ranged over by a, b, c:

$$a \quad := \quad x \mid \lambda x.a \mid a\,b.$$

  We are always speaking up to α-conversion.
- One-step β-reduction is defined by:

$$(\lambda u.b)\,a > [a/u]b \quad (\beta)$$

$$\frac{a > a'}{\lambda u.a > \lambda u.a'}\,(\xi) \qquad \frac{a > a'}{a\,b > a'\,b}\,\textit{(appl)} \qquad \frac{b > b'}{a\,b > a\,b'}\,\textit{(appr)}$$

- **Substitution lemma.**
  If $a >^* b$, then $[a/x]c >^* [b/x]c$ and $[c/x]a >^* [c/x]b$.
- A term, a, is in $>^*$**-normal form** iff a has no $>$-reductions.

Church-Rosser (CR) theorem

- The CR theorem states that $>$ is confluent, i.e. $dp(>^*)$.
- The same lemma holds for combinatory reduction for *s, k* combinators. The same proof idea works.
- **corollary:** normal forms are unique: if $a >^* b$, $a >^* c$, and b, c are both in normal form, then $b = c$ (up to α-conversion).
  - **proof** By diamond property, b, c reduce to a common term, d. But b, c are both in normal form, so $b = d = c$. □
- CR does *not* say that every term has a normal form, or that if one reduction sequence reaches a normal form then every reduction sequence reaches a normal form.

Strip lemmas

For any $\to \in$ *Rel2*, $dp(\to) \;\Rightarrow\; dp(\to^*)$ by the diagram chase:

$$
\begin{tikzcd}
a \arrow[r] \arrow[d] \arrow[rrr, bend left, "*"] \arrow[ddd, bend right, "*"'] & \cdot \arrow[r] \arrow[d] & \cdot \arrow[r, dotted] \arrow[d] & c \arrow[d] \\
\cdot \arrow[r] \arrow[d] & \cdot \arrow[r] \arrow[d] & \cdot \arrow[r, dotted] \arrow[d] & \cdot \arrow[d] \\
\cdot \arrow[r] \arrow[d, dotted] & \cdot \arrow[r] \arrow[d, dotted] & \cdot \arrow[r, dotted] \arrow[d, dotted, no head] & \cdot \arrow[d, dotted] \\
b \arrow[r] & \cdot \arrow[r] & \cdot \arrow[r, dotted] & d
\end{tikzcd}
$$

This is really double induction:

- first along the top, showing that every rectangle of height 1 commutes (called the *strip lemma*),
- then along the side, showing that every rectangle commutes.
- See my paper to get same result with a single induction.

Thus, if we had $dp(>)$, we would be finishesd; but that is not the case. Two things go wrong.

What is wrong with $>$?

(1) $>$ can forget parts of a term, but is not reflexive

$$
\begin{tikzcd}
 & (\lambda x.y)\,((\lambda x.x)z) \arrow[dl] \arrow[dr] & \\
(\lambda x.y)\,z \arrow[dr] & & y \\
 & y &
\end{tikzcd}
$$

What is wrong with $>$?

(2) $>$ can copy parts of a term, but is not parallel

$$\begin{tikzcd}
& (\lambda x.xx)\,((\lambda x.y)z) \arrow[dl] \arrow[dr] & \\
((\lambda x.y)z)\,((\lambda x.y)z) & & (\lambda x.xx)\,y \arrow[dl] \\
& y\,y &
\end{tikzcd}$$

An Outline of the Proof

The idea of Tait and Martin-Löf: define a relation of **parallel reduction**, $\gg$, that is both reflexive and parallel.

1. The subtle part is showing $dp(\gg)$.
   - I present an improvement (due to Takahashi) of the Tait–Martin-Löf proof.
2. The easy part is showing $dp(\gg)$ implies $dp(\gg^*)$.
   - This is the strip lemma we saw above.
3. Showing $a \gg^* b$ iff $a >^* b$ (hence $dp(>^*)$, our goal).
   - This is usually considered trivial, but in fact the names of variables are problematic.
   - We skip the problematic details.

Parallel Reduction, $\gg$

pr-refl $\quad x \gg x$

pr-β $\quad \dfrac{a \gg a' \qquad b \gg b'}{(\lambda u.a)\, b \gg [b'/u]a'}$

pr-ξ $\quad \dfrac{a \gg a'}{\lambda u.a \gg \lambda u.a'}$

pr-app $\quad \dfrac{a \gg a' \qquad b \gg b'}{a\, b \gg a'\, b'}$

- $\gg$ is *reflexive* by rule *pr-refl*.
  - General reflexivity, $a \gg a$, is derivable.
- $\gg$ is *parallel*: rules *pr-β* and *pr-app* allow reduction in both subterms.
- Non-deterministic choice of which rule to apply to a redex, *pr-β* or *pr-app*.

Intuition about parallel reduction ($\gg$)

- Any 2 redexes in a term are either disjoint, or one is contained in the other.
  - All redexes are subterms, and subterms have this property.
  - This holds for combinator terms as well as λ-terms.
- We can unambiguously mark each redex in a term, say by putting a unique identifier on its outer application.
- $\gg$ allows contracting any subset of the marked redexes
  - Contracting may discard some redexes, and may copy some redexes (copy the marks with the redexes).
  - Contracting may also create brand new unmarked redexes.
  - The redexes with a particular mark left after a $\gg$-step are called **residuals** of the original redex with that mark.

Intuition about the proof of CR

- To prove $dp(\gg)$, we are given 2 reductions, $a \gg b$, $a \gg c$, contracting different sets of marked redexes, say $\mathcal{M}_b$ and $\mathcal{M}_c$
- To complete the diamond, just contract the necessary marked redexes.
  - In b, (resp. c), contract any redexes from $\mathcal{M}_c$ (resp. $\mathcal{M}_b$) that are left.
  - Any redexes not in $\mathcal{M}_b$ or $\mathcal{M}_c$ can be ignored.
  - Any new (unmarked) redexes can be ignored.
- At the end the same set of redexes will have been contracted along both reduction paths, so they will end at the same term.

$$\begin{tikzcd} & a \arrow[dl, two heads, "\mathcal{M}_b"'] \arrow[dr, two heads, "\mathcal{M}_c"] & \\ b \arrow[dr, two heads, "\mathcal{M}_c"'] & & c \arrow[dl, two heads, "\mathcal{M}_b"] \\ & d & \end{tikzcd}$$

$dp(\gg)$: Parallel Reduction has the diamond property

- We could actually mark the redexes (Huet 1994) ...
- usual proof keeps track implicitly of which redexes are contracted.

lemma (CR): $\forall a, b, c \,.\, a \gg b \ \wedge \ a \gg c \ \Rightarrow \ \exists d \,.\, b \gg d \ \wedge \ c \gg d$.
proof By "induction on the structure of *a*" (Shankar 1988):

- *a* is a variable (trivial)
- *a* is a lambda (easy)
- *a* is an application; 5 subcases
  1. *a* is not a redex
  2. *a* is a redex, and is only contracted in the reduction $a \gg b$
  3. *a* is a redex, and is only contracted in the reduction $a \gg c$
  4. *a* is a redex, and is contracted in both reductions
  5. *a* is a redex, and is not contracted in either reduction

. . . proving $dp(\gg)$

For example in case 4, suppose $a = (\lambda x.a_l)\, a_r$ and $a \gg b$ (respectively $a \gg c$) by two instances of *pr-beta*

$$\frac{a_l \gg a_l^b \quad a_r \gg a_r^b}{(\lambda x.a_l)\, a_r \gg [a_r^b/x]a_l^b} \qquad \frac{a_l \gg a_l^c \quad a_r \gg a_r^c}{(\lambda x.a_l)\, a_r \gg [a_r^c/x]a_l^c}$$

By the two induction hypotheses we have the diagrams

$$\begin{tikzcd}
& a_l \arrow[dl] \arrow[dr] & \\
a_l^b \arrow[dr, dotted] \arrow[rr, phantom, "\text{left IH}"] & & a_l^c \arrow[dl, dotted] \\
& d_l &
\end{tikzcd}
\qquad
\begin{tikzcd}
& a_r \arrow[dl] \arrow[dr] & \\
a_r^b \arrow[dr, dotted] \arrow[rr, phantom, "\text{right IH}"] & & a_r^c \arrow[dl, dotted] \\
& d_r &
\end{tikzcd}$$

By the substitution lemma (for $\gg$) we have $[a_r^b/x]a_l^b \gg [d_r/x]d_l$ and $[a_r^c/x]a_l^c \gg [d_r/x]d_l$ as required.

Another Proof of $dp(\gg)$

- By "simultaneous induction on the structure of the reductions" $a \gg b$ and $a \gg c$ (Pfenning 1992).
- The argument goes as above, but the two uses of IH diagrammed above are justified because the derivations of
  - $a_l \gg a_l^b$ and $a_l \gg a_l^c$
  - respectively $a_r \gg a_r^b$ and $a_r \gg a_r^c$

  are subderivations of the given derivation pair, $a \gg b$ and $a \gg c$.

These proofs are too fine!

- They analyze what redexes are contracted in the two given reductions, $a \gg b$ and $a \gg c$;
- they close the diamond by contracting only the redexes that are necessary to bring b and c together.

Why be so careful?

- In b, contracting *all* the marked redexes left
  - without regard to what redexes were contracted in $a \gg c$

  will have the same effect as if we just contracted all the redexes in a to start with.
- The same is true of c.
- This will close the diamond, although it may contract some redexes that were not necessary to do so.
- the bottom of the diamond will be the "maximum $\gg$-step from a".

A Coarser Proof of $dp(\gg)$ (Takahashi)

- Taking the "maximum" parallel reduction step that contracts all redexes in *a*, we can close any triangle

$$\begin{tikzcd} & a \arrow[dl, "\gg"'] \arrow[dd, "\text{maximum parallel reduction}"] \\ b \arrow[dr, dotted, "\gg"'] & \\ & d \end{tikzcd}$$

  by contracting all the residuals in *b* of redexes in *a*.
- Then we can complete any diamond by closing the left and right triangles independently.

Define a new relation, called **complete development**, $\ggg$, that contracts all the redexes in a term.

Complete Developments

cd-var $\quad x \ggg x$

cd-β $\quad \dfrac{a \ggg a' \qquad b \ggg b'}{(\lambda u.a)\, b \ggg [b'/u]a'}$

cd-ξ $\quad \dfrac{a \ggg a'}{\lambda u.a \ggg \lambda u.a'}$

cd-app $\quad \dfrac{a \ggg a' \qquad b \ggg b' \qquad a \text{ is not an abstraction}}{a\, b \ggg a'\, b'}$

$\ggg$ is "the same as $\gg$ but goes farther":

- The non deterministic choice in $\gg$ to use *pr-β* or *pr-app* on a redex is removed.
- $\ggg$ contracts every redex (but not newly created ones).

$\ggg$ is the "maximum parallel reduction"

- **lemma** ($\ggg$ exists) Every term has a complete development: $\forall a\, \exists d\, .\ a \ggg d.$
    - **proof** Easy structural induction on a: in every case exactly one rule applies. □
- In fact complete development is unique, but we don't need that fact.
- **lemma** (triangle) $\ggg$ closes any $\gg$ triangle:

$$\forall a, d, b\ .\ a \ggg d \ \wedge\ a \gg b \ \Rightarrow\ b \gg d.$$

$$\begin{tikzcd} & a \arrow[dl, "\gg"'] \arrow[dd, "\ggg"] \\ b \arrow[dr, dashed, "\gg"'] & \\ & d \end{tikzcd}$$

Proof that $a \ggg d \land a \gg b \Rightarrow b \gg d$

Structural induction on $a \ggg d$.

- Consider the case where $a = (\lambda u.a_l)\, a_r \ggg [a_r^d/u]a_l^d = d$ because

$$\frac{a_l \ggg a_l^d \qquad a_r \ggg a_r^d}{(\lambda u.a_l)\, a_r \ggg [a_r^d/u]a_l^d}\ (cd\text{-}\beta)$$

- Two subcases for the 2 possible $\gg$-steps from $a = (\lambda u.a_l)\, a_r$:
  - First subcase: $(\lambda u.a_l)\, a_r$ reduces by $pr\text{-}\beta$:

$$\frac{a_l \gg a_l^b \qquad a_r \gg a_r^b}{(\lambda u.a_l)\, a_r \gg [a_r^b/u]a_l^b}\ (pr\text{-}\beta)$$

    - By IH we have $a_l^b \gg a_l^d$ and $a_r^b \gg a_r^d$,
    - by substitution lemma, $[a_r^b/u]a_l^b \gg [a_r^d/u]a_l^d$ as required.

$\ldots \quad a \ggg d \wedge a \gg b \Rightarrow b \gg d$

...proof continued

- Still the case where $a = (\lambda u.a_l)\, a_r \ggg [a_r^d/u]a_l^d = d$ because

$$\frac{a_l \ggg a_l^d \qquad a_r \ggg a_r^d}{(\lambda u.a_l)\, a_r \ggg [a_r^d/u]a_l^d}\ (cd\text{-}\beta)$$

  - Second subcase: $(\lambda u.a_l)\, a_r$ reduces by *pr-app*:

$$\frac{\dfrac{a_l \gg a_l^b}{\lambda u.a_l \gg \lambda u.a_l^b}\ \textit{pr-lda} \qquad a_r \gg a_r^b}{(\lambda u.a_l)\, a_r \gg (\lambda u.a_l^b)\, a_r^b}\ \textit{pr-app}$$

    - By IH we have $a_l^b \gg a_l^d$ and $a_r^b \gg a_r^d$,
    - by rule *pr-*β, $(\lambda u.a_l^b)\, a_r^b \gg [a_r^d/u]a_l^d$ as required. $\square$

Church–Rosser theorem

- **lemma** $dp(\gg)$: $a \gg b \wedge a \gg c \Rightarrow \exists d \,.\, b \gg d \wedge c \gg d$.
  - **proof** Let d be s.t. $a \ggg d$ ($\ggg$ exists). $b \gg d$ and $c \gg d$ by (triangle). □
- This proof is *coarser* than the standard one:
  - It treats less cases, by using a deterministic relation $\ggg$ instead of figuring out which redexes must be contracted by $\gg$.
  - It produces a worse program to compute d, contracting more redexes than necessary.
- **corollary** $dp(\gg^*)$
  - **proof** By a strip lemma diagram chase. □
- **corollary** $a \gg^* b$ iff $a >^* b$; hence $dp(>^*)$.
  - **proof** ($\Leftarrow$) $a > b \Rightarrow a \gg b$ is trivial, as every $>$-step is also a $\gg$-step.
  - ($\Rightarrow$) $a \gg b \Rightarrow a >^* b$ is proved by induction on the derivation of $a \gg b$. □

More about $\ggg$

- $\ggg$ is not just a reduction relation, it is deterministic, i.e. $\ggg$ is a **strategy**.
- **lemma** $\ggg^*$ is **cofinal** for $>^*$, i.e.

$$\forall a, b \,.\, a >^* b \;\Rightarrow\; \exists d \,.\, a \ggg^* d \;\wedge\; b >^* d$$

  - **proof** Easy from what we have proved above. $\square$
- **lemma** $\ggg^*$ is normalizing: if a has a normal form, then $\ggg^*$ deterministically finds it.
  - **proof** If a_n is the normal form of a, then $a >^* a_n$, so for some d, $a \ggg^* d$ and $a_n >^* d$. But a_n is normal, so $a = d$. $\square$
- $\ggg^*$ is an easier normalizing strategy to reason about than other such strategies (call-by-need, call-by-name, ...)

Conclusion

- Very well known proofs can be made more beautiful.
- Look for alternative inductive definitions that are easier to reason with.
  - In this search, try to eliminate unnecessary non-determinism in definitions.
- For program extraction must pay attention to the algorithmic content of proofs.