Abstract Lambda Calculus

Gert Smolka, Saarland University

November 21, 2019

We formalize λ-calculus using de Bruijn terms and show confluence of reduction using the TMT method. We postpone the details of substitution by working with an abstract function for β-reduction. For call-by-value reduction we show uniform confluence for every β function.

1 Abstract Call-By-Value Calculus

We employ de Bruijn terms as follows:

$$s, t, u, v \ ::= \ x \mid st \mid \lambda s \qquad (x \in \mathsf{N})$$

Figure 1 defines the reduction relation of an abstract version of the call-by-value lambda calculus we call **abstract** $\lambda\beta_v$. The rule for β-reduction assumes a function β that given two terms yields a term. The function β hides the subtleties of substitution.

It is not difficult to show that abstract $\lambda\beta_v$ is uniformly confluent. The informal reason for uniform confluence is the fact that reduction takes place only at the leaves of a binary tree (obtained by applications st), and that there is only one reduction possible per leaf. Thus given $s \succ_v t_1$ and $s \succ_v t_2$, t_1 and t_2 are obtained by reduction of the same leaf, in which case they are equal, or t_1 and t_2 are obtained by reduction of different leaves, in which case they can be joined by reducing in each case the leaf from the other reduction.

Fact 1 Abstract call-by-value reduction $\succ_v$ is uniformly confluent for every function β.

2 Abstract Lambda Beta

Figure 2 defines the reduction relation of an abstract version of the lambda calculus we call **abstract** $\lambda\beta$. The rule for β-reduction assumes a function β that given two terms yields a term.

$$\frac{\text{abstraction } t}{(\lambda s)t \succ_v \beta st} \qquad \frac{s \succ_v s'}{st \succ_v s't} \qquad \frac{t \succ_v t'}{st \succ_v st'}$$

Figure 1: Definition of abstract call-by-value reduction

$$\frac{}{(\lambda s)t \succ \beta st} \qquad \frac{s \succ s'}{st \succ s't} \qquad \frac{t \succ t'}{st \succ st'} \qquad \frac{s \succ s'}{\lambda s \succ \lambda s'}$$

Figure 2: Definition of reduction for abstract $\lambda\beta$

A binary relation R on terms is **compatible** (with the term structure) if it satisfies the following rules:

$$\frac{Rss'}{R(st)(s't)} \qquad \frac{Rtt'}{R(st)(st')} \qquad \frac{Rss'}{R(\lambda x.s)(\lambda x.s')}$$

Fact 2 (Compatibility)

1. $s \succ t$ is compatible.
2. $s \succ^* t$ is compatible.
3. If $s \succ^* s'$ and $t \succ^* t'$, then $st \succ^* s't'$.

Proof Claim 1 holds by definition. Claims 2 follows by star induction and claim 1. Claim 3 follows from claim 2. ∎

3 Confluence of Abstract Lambda Beta

Confluence of abstract $\lambda\beta$ can be shown with the TMT method. We will need one natural assumption about β to show confluence of abstract $\lambda\beta$.

The key idea is the definition of an intermediate relation $\succ \subseteq \gg \subseteq \succ^*$ known as **parallel reduction**. The definitions of parallel reduction $\gg$ and the accompanying Takahashi function ρ appear in Figure 3.

Fact 3 (Reflexivity) $s \gg s$.

Proof By induction on s. ∎

Fact 4 (Sandwich) $\succ \subseteq \gg \subseteq \succ^*$.

$$\frac{s \gg s' \qquad t \gg t'}{(\lambda s)t \gg \beta s' t'} \qquad \frac{}{x \gg x} \qquad \frac{s \gg s' \qquad t \gg t'}{st \gg s't'} \qquad \frac{s \gg s'}{\lambda s \gg \lambda s'}$$

$$\begin{aligned} \rho((\lambda s)t) &= \beta(\rho s)(\rho t) \\ \rho(st) &= (\rho s)(\rho t) \qquad \text{if } s \text{ not an abstraction} \\ \rho(\lambda s) &= \lambda(\rho s) \\ \rho(x) &= x \end{aligned}$$

Figure 3: Parallel reduction and Takahashi function for abstract $\lambda\beta$

Proof The first inclusion follows by induction on $\succ$ using Fact 3. The second inclusion follows by induction on $\gg$ using compatibility (Fact 2). ∎

We say that β is **compatible** if $\beta st \gg \beta s't'$ whenever $s \gg s'$ and $t \gg t'$.

Fact 5 Let β be compatible. Then ρ is a Takahashi function for $\gg$.

Proof Let $s \gg t$. We prove $t \gg \rho s$ by induction on $s \gg t$. The case for the β-rule follows with the compatibility of β. No other case needs the compatibility assumption. The cases for variables and abstractions are straightforward. The final case is for the compatibility rule for applications. If $s = xs'$ or $s = s_1 s_2 s_3$, the claim follows with the inductive hypotheses. Otherwise, $s = (\lambda s_1)s_2$, $t = (\lambda s_1')s_2'$, $s_1 \gg s_1'$, and $s_2 \gg s_2'$. We need to show $(\lambda s_1')s_2' \gg \beta(\rho s_1)(\rho s_2)$, which follows with the induction hypotheses $s_1' \gg \rho s_1$ and $s_2' \gg \rho s_2$. ∎

Theorem 6 Let β be compatible. Then $\succ$ is confluent and ρ is a reduction function for $\succ$.

Proof Follows with the TMT theorem using Facts 5 and 4. ∎

Exercise 7 Give an example that shows that parallel reduction is not transitive.

Exercise 8 (Confluence of SK-Calculus) The SK-calculus employs applicative terms

$$s, t ::= x \mid K \mid S \mid st \qquad (x : \mathsf{N})$$

obtained with variables, two constants K and S, and application. Reduction in the SK-calculus is defined as follows:

$$\frac{}{Kst \succ s} \qquad \frac{}{Sstu \succ su(tu)} \qquad \frac{s \succ s'}{st \succ s't} \qquad \frac{t \succ t'}{st \succ st'}$$

$$\frac{}{(\lambda s)t \equiv \beta st} \qquad \frac{s \equiv s' \qquad t \equiv t'}{st \equiv s't'} \qquad \frac{s \equiv s'}{\lambda s \equiv \lambda s'} \qquad \frac{}{s \equiv s} \qquad \frac{s \equiv t}{t \equiv s} \qquad \frac{s \equiv t \qquad t \equiv u}{s \equiv u}$$

Figure 4: Equivalence rules for de Bruijn terms

Prove that reduction in the SK-calculus is confluent. Use the TMT method with a suitably defined parallel reduction.

Some background. SK is also known as combinatory logic and may be seen as a subsystem of $\lambda\beta$ where $K = \lambda xy.x$ and $S = \lambda fgx.(fx)(gx)$. SK can express all computable functions and has been extensively studied by logicians.

4 Equivalence

Recall that we have defined the equivalence relation accompanying a reduction relation R as $R^{\leftrightarrow *}$. This definition was useful for the proof that confluence of R entails the Church-Rosser property.

As it comes to equivalence of terms, we would hope that $\succ^{\leftrightarrow *}$ satisfies the equivalence rules shown in Figure 4. This is in fact the case.

Fact 9 Equivalence of terms $\succ^{\leftrightarrow *}$ satisfies the rules in Figure 4.

Proof Reflexivity and transitivity are clear from the definition with star. Since star preserves symmetry, we also have symmetry. As it comes to compatibility, we first show that R^* is compatible if R is compatible. It now suffices to show that $\succ^{\leftrightarrow}$ is compatible, which follows from the fact that $\succ$ is compatible by definition and that $\leftrightarrow$ preserves compatibility. ∎

We may ask whether the equivalence rules in Figure 4 suffice for proving all term equivalences. This is in fact the case and can be argued in three steps

$$\begin{aligned} s \succ t &\to s \equiv t \\ s \succ^{\leftrightarrow} t &\to s \equiv t \\ s \succ^{\leftrightarrow *} t &\to s \equiv t \end{aligned}$$

where $\equiv$ is any relation satisfying the rules in Figure 4. The first implication follows by induction on $s \succ t$, and the third implication follows by star induction. We summarize our results as follows.

Fact 10 Term equivalence $s \succ^{\leftrightarrow *} t$ agrees with the relation $s \equiv t$ defined inductively with the rules in Figure 4.

5 Historical Remarks

The notion of parallel reduction goes back to Curry and Feys [1]. The inductive definition of parallel reduction and its use for confluence proofs is due to Tait (1965) and Martin-Löf (1971) (see Hindley and Seldin [2], Appendix A2). Takahashi functions originated with Takahashi's [3] confluence proof for $\lambda\beta$.

References

[1] Haskell B. Curry and Robert Feys. *Combinatory Logic, Volume I.* North-Holland Publishing Company, 1958.

[2] J. Roger Hindley and Jonathan P. Seldin. *Lambda-Calculus and Combinators, an Introduction.* Cambridge University Press, 2008.

[3] Masako Takahashi. Parallel reductions in λ-calculus. *Inf. Comput.*, 118(1):120–127, 1995.