Abstract. This paper presents a theorem prover for a combination of constructive first-order logic and the λ-calculus. The paper presents the basic theorem prover, which is an extension of [6]'s model generation theorem prover for first-order logic, and considers issues relating to the compile-time optimisations that are often used with first-order theorem provers.

1 A Constructive Intensional Logic

For various reasons, the idea of a language which allows you to construct abstractions and apply them to terms, as in the λ-calculus, and to combine these operations with the truth functional connectives of predicate logic, is extremely tempting. It is well known, however, that simply adding the λ-calculus and predicate logic together opens the way to the paradoxes of negative self-reference – the Liar, Russell’s set, and so on.

The classical way out of this is to place restrictions on what can be said [11,8,3]. [9] approaches the matter by allowing you to say whatever you want, but then placing constraints on what can be proved. The current paper follows this general approach, but uses entirely different constraints.

Turner takes a classical treatment of first-order logic and adds λ-abstraction and β-reduction to it (or at any rate, operations which look extremely like λ-abstraction and β-reduction). In order to avoid the paradoxes, however, he constrains the circumstances under which you are allowed to perform λ-abstraction. The constraints he chooses are enough to make the underlying logic consistent (in other words, to avoid the paradoxes), and makes the paradoxes UNSTABLE [2]. The current paper takes a constructive treatment of first-order logic, allows unrestricted use of both λ-abstraction and β-reduction, but avoids the paradoxes by placing constraints on the assumptions that can be used in a well-founded proof.

The logic which we will use, which we will call Λ(C) for constructive λ-calculus, extends first-order logic as follows:

Λ(C)-1 If A is a formula of first-order logic then it is a formula of Λ(C), and if t is a term of first-order logic then it is a term of Λ(C).
$\Lambda(C)$-2 If $A$ is a formula of $\Lambda(C)$, possibly including free occurrences of $x$, then

$\lambda x A$ is a term of $\Lambda(C)$.

$\Lambda(C)$-3 If $t$ and $t'$ are terms of $\Lambda(C)$ then $t.t'$ is a formula.

The proof theory for $\Lambda(C)$ is obtained by adding the rules in Fig. 1 to a 
standard set of natural deduction rules, which I will refer to as ND (these rules 
are omitted here for space reasons, but any standard text on constructive logic 
will provide such a set, e.g. [10]).

\begin{enumerate}
\item[(\lambda\text{-intro})] $\alpha \vdash \ldots, A, \ldots \Rightarrow \alpha \vdash \ldots, (\lambda x A').t, \ldots$
\item[(\ldots, A, \ldots \Rightarrow \alpha \vdash \ldots, (\lambda x A').t, \ldots] is any formula containing $A$ as a subformula or a term, 
and $\ldots, A, \ldots$, the formula that is obtained from $\ldots, A, \ldots$ by 
replacing 0 or more instances of $t$ in $A$ by $x$.
\item[(\lambda\text{-elim})] $\alpha \vdash \ldots, (\lambda x A).t, \ldots \Rightarrow \alpha \vdash \ldots, A_{t/x}, \ldots$
\end{enumerate}

\textbf{Fig. 1.} Natural deduction rules for $\Lambda(C)$

(\lambda\text{-intro}) and (\lambda\text{-elim}) add $\lambda$-abstraction and $\beta$-reduction to ND. Theorem 1 shows that we can do this without introducing proofs of $\bot$.

\textbf{Theorem 1.} Soundness of $\Lambda(C)$

If there is no proof of $\bot$ from $\alpha$ using ND then any proof of $\bot$ from $\alpha$ using all 
the rules of $\Lambda(C)$ introduces some irreducible instance of $(\lambda x A).t$.

\textbf{Proof.}

Suppose that $\alpha_0 \vdash A_0, \alpha_1 \vdash A_1, \ldots, \alpha_n \vdash \bot$ is a proof of $\bot$ from $\alpha_0$ using the 
rules $\Lambda(C)$, where $\alpha_0$ contains no irreducible $\lambda$-applications and there is no proof 
of $\bot$ from $\alpha_0$ just using ND; and that there is no proof of $\bot$ from any set $\beta$ which 
also satisfies the conditions but which contains fewer applications of $\lambda$-elim and $\lambda$-intro.

Consider the first use in this proof of $\bot$ from $\alpha_0$ of either (i) $\lambda$-elim to change 
some formula $(\lambda x A).t$ into $A_{t/x}$, or (ii) $\lambda$-intro to change some formula $A_{t/x}$ into 
$(\lambda x A).t$ (there must be one, since otherwise $\alpha_0$ would have supported a proof 
of $\bot$ from ND alone). In case (i) we can obtain a proof of $\bot$ from $\alpha \cup A_{t/x}$, 
and in case (ii) we can obtain one from $\alpha \cup (\lambda x A).t$, each of which omits the 
relevant step, and hence involves fewer applications of these rules, contradicting 
the assumption (note that $(\lambda x A).t$ is irreducible iff. $A_{t/x}$ is, so that the first step 
which adds either of these to $\alpha$ will not introduce an irreducible formula unless 
there was already one there) \hfill \square

The point of this theorem is that any proof of $\bot$ from a set which is consistent 
under the first-order rules must introduce some irreducible formula (since other-
wise every subproof would satisfy the conditions of the theorem). If we take 
the constructive view of $\lambda$-applications as promissory notes for proofs, or for

\footnote{ $(\lambda x A).t$ is irreducible if there is no sequence of applications of $\lambda$-elim which will 
produce a term with no occurrences at all of $(\lambda y B).s$.}