Fast Normalization in the HOL Theorem Prover

Joe Hurd*
Computer Laboratory
University of Cambridge
joe.hurd@cl.cam.ac.uk

1 March 2002

1 Introduction

The 'LCF design' inherited by the HOL theorem prover [1] allows users to write ML functions encoding arbitrary patterns of logical deduction. Such ML functions are called derived rules,[1] because the only way that they can create theorems is by composing a sequence of existing rules. The initial rules, implemented in a small logical kernel, are the primitive inference rules of higher-order logic. This design philosophy is called fully-expansive proof, because every derived rule can be 'fully expanded' to a (long) sequence of primitive inferences.

Example: Harrison's HOL implementation of the model elimination procedure is a derived rule. It takes a goal term, performs a proof search, and then executes the successful proof as a sequence of primitive inferences. □

We are interested in creating efficient derived rules for normalizing terms. Converting terms to conjunctive normal form (CNF) is necessary to apply many first-order proof procedures, and some decision procedures for Presburger arithmetic transform terms to disjunctive normal form as part of their operation. Our present focus is on converting terms to definitional CNF: a normal form similar to CNF in which the result formulas are only linearly larger than the input formulas. An upper bound on the result size is essential when normalizing large propositional formulas (such as those generated by hardware verification) for input to satisfiability (SAT) solvers. In our experiments, we use Gordon's `HolSatLib`[2] library for `hol98`. It provides a harness for invoking SAT solvers on HOL terms, and currently supports `SATO`, `GRASP` and `zCHAFF`.

Example: To prove that the boolean 'exclusive or' operation is associative, we

*Supported by EPSRC project GR/R27105/01.
[1] Or tactics, depending on the context in which they are used.
[2] `http://www.cl.cam.ac.uk/users/mjcg/HolSatLib/`

first convert the negation of the goal to definitional CNF:

$$\begin{aligned}\vdash\ & \neg((\neg(p = \neg(q = r)) = \neg(\neg(p = q) = r))) = \\ & \exists\, v_0, v_1, v_2, v_3, v_4. \\ & \quad (v_4 \lor v_1 \lor v_3) \land (v_4 \lor \neg v_1 \lor \neg v_3) \land (v_1 \lor \neg v_3 \lor \neg v_4) \land (v_3 \lor \neg v_1 \lor \neg v_4) \land \\ & \quad (v_3 \lor v_2 \lor r) \land (v_3 \lor \neg v_2 \lor \neg r) \land (v_2 \lor \neg r \lor \neg v_3) \land (r \lor \neg v_2 \lor \neg v_3) \land \\ & \quad (v_2 \lor p \lor \neg q) \land (v_2 \lor \neg p \lor q) \land (p \lor q \lor \neg v_2) \land (\neg q \lor \neg p \lor \neg v_2) \land \\ & \quad (v_1 \lor p \lor v_0) \land (v_1 \lor \neg p \lor \neg v_0) \land (p \lor \neg v_0 \lor \neg v_1) \land (v_0 \lor \neg p \lor \neg v_1) \land \\ & \quad (v_0 \lor q \lor r) \land (v_0 \lor \neg q \lor \neg r) \land (q \lor \neg r \lor \neg v_0) \land (r \lor \neg q \lor \neg v_0) \land v_4\end{aligned}$$

Next we strip off the existential quantifiers (the 'definitions' of definitional CNF) from the RHS of this equation, and invoke `HolSatLib` on the resulting term. The HOL term is then converted to DIMACS format, sent to a SAT solver, and, if no satisfiable assignments are found, the negation of the term is asserted as a HOL theorem (with a 'SAT solver' oracle tag[3]). Finally, a couple more primitive inferences suffice to derive our original goal as a theorem. □

As can be observed, the above method of proving propositional formulas involves a conversion to definitional CNF. It is relatively easy to generate the desired term, and so we can simply create the conversion theorem with a 'normalization' oracle tag. However, it is interesting to see how efficiently we can construct the fully-expansive proof of the conversion theorem. We could naively try to mirror the definitional CNF algorithm using primitive inferences, but large intermediate terms make this algorithm is quadratic in the size of the input term. Instead, we use a technique of Harrison [2] using variable vectors to create a novel algorithm that runs in (nearly) linear time. On our `ADD4` example term,[4] this uses more primitive inferences than the naive method, but small terms at each intermediate point mean that the overall time taken is lower.

Operation on the `ADD4` Term	**Time (s)**	**Infs.**
Definitional NNF	10.610	7677
Oracle version of definitional CNF	0.390	0
Naive definitional CNF	122.620	12034
Definitional CNF using variable vectors	28.800	238258
Applying the `zCHAFF` solver	4.680	0

References

[1] M. J. C. Gordon and T. F. Melham. *Introduction to HOL (A theorem-proving environment for higher order logic).* Cambridge University Press, 1993.

[2] John Harrison. Binary decision diagrams as a HOL derived rule. *The Computer Journal*, 38:162–170, 1995.

[3] Theorem tags are propagated by the primitive inferences, and so for each theorem it is easy to discover the tools that were used as oracles in the proof.

[4] The `ADD4` term arose from hardware verification, and contains 1111 atoms.