Automated Proof Search in Gödel-Löb Provability Logic

Rajeev Goré and Jack Kelly
The Australian National University, Canberra, Australia

The Gödel-Löb logic G is a propositional normal modal logic where $\Box A$ can be interpreted as "A is provable in Peano arithmetic". In addition to the axioms of K, logic G also has the axiom $\Box(\Box A \rightarrow A) \rightarrow \Box A$. G is characterised by transitive and well-founded frames. If $\neg$, $\Box$ and $\vee$ are treated as primitive, with implication and $\Diamond$ defined in terms of these, the tableau rules shown below give a sound and complete tableau calculus for G:

$$(Id)\frac{A; \neg A; X}{\bot} \qquad (\neg\vee)\frac{\neg(A \vee B); X}{\neg A; \neg B; X} \qquad (\vee)\frac{A \vee B; X}{A; X | B; X} \qquad (G)\frac{\neg\Box A; \Box X; Z}{\neg A; X; \Box X; \Box A}$$

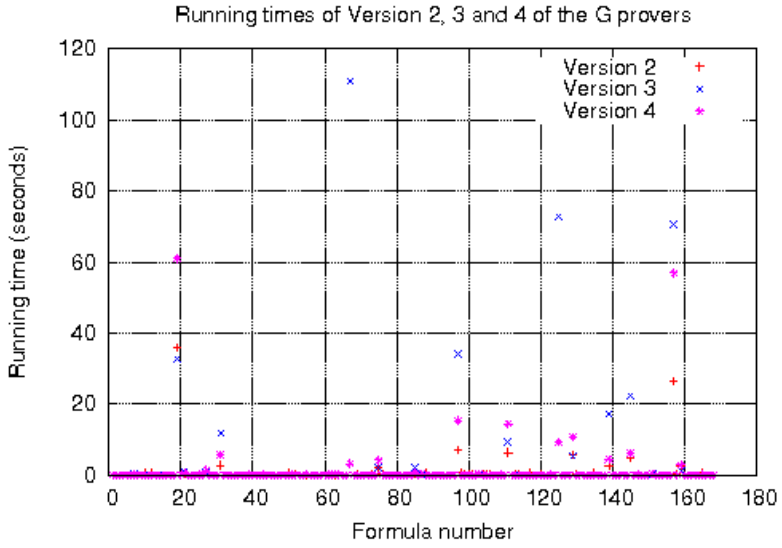
Proof search in this calculus terminates if the following procedure is applied repeatedly: the first three rules are applied until they are no longer applicable, and the (G) rule is applied afterwards. Intuitively, a $\Diamond\neg A$ (i.e. $\neg\Box A$) creates a $\Box A$ so that a subsequent application of the (G)-rule on $\Diamond\neg A$ causes closure via $A; \neg A$. We therefore tried to create a theorem prover for G using the Tableau Work Bench, a framework for building automated theorem provers available from `http://twb.rsise.anu.edu.au`.

As is traditional in automated theorem proving, we assumed that all formulae were first put into negation normal form (nnf) where implications are translated away and all negations are distributed to atomic level. Thus the (Id)-rule becomes $p; \neg p; X$. The TWB is designed to have its input closely resemble formally written tableau rules so the nnf variant of tableau rule (G) as shown below left is represented in the TWB as shown below right:

$$(Gnnf)\frac{\Diamond A \ ; \ \Box X \ ; \ Z}{A \ ; \ X \ ; \ \Box X \ ; \ nnf(\Box\neg A)}$$

```
RULE Gnnf
  { Dia A } ; Box X ; Z
 ---------------------------------
    A ; X ; Box X ; nnf(Box Neg A)
END
```

To our surprise, Version 1 as shown above, did not terminate due to a missing fairness constraint, with the culprit being our nnf assumption. Version 2 used $\Box\neg A$ instead of $nnf(\Box\neg A)$, breaking the nnf-format in a deliberate way, and also used $A; \neg A$ as the closure condition in the (Id) rule. Version 3 kept the new (Id) rule, replaced $\Box\neg A$ with $\Box\neg\Diamond A$ to try and close branches with fewer rule applications, and resulted in a small speed increase for some input formulae but a severe increase in running time for many others. As a hybrid, Version 4 kept both $\Box\neg A$ and $\Box\neg\Diamond A$ in the denominator, and improved on the execution times for some inputs over Version 3. For input formulae that had similar execution times when evaluated by versions two and three, this hybrid prover took around twice as long. The running times of the various provers for random input formulae are summarised by the graph below.



We analyse the cause of the divergence between the expected terminating behaviour of the theoretical calculus and the actual implementations. We also report on the running times of the various provers we built. The main logical contribution is the following theorem, which is not trivial:

Theorem 1 All nnf-versions of our calculus, except the first one, are sound, complete and terminating.