Scala Project

Maximilien Colange

The goal of this project is to build a simple theorem prover. We focus on propositional logic, which is decidable (i.e. for every formula we can decide whether it is valid or not) and complete (i.e. every valid formula is provable).

We will use a variant of Gentzen's sequent calculus.

1 Logics

Let $\mathcal{X}$ be a set of propositional variables. The grammar of formulae is as follows:

$$\phi ::= \top \mid \bot \mid x \mid \phi \wedge \phi \mid \phi \vee \phi \mid \neg\phi$$

where $x \in \mathcal{X}$. This means that a Boolean formula is either a Boolean constant ($\top$ or $\bot$), or a Boolean variable, or a Boolean combination of Boolean formulae.

We say that a formula ϕ is *valid* if it is true for every valuation of its variables. A valid formula is also called a theorem.

One possible, yet naive, approach to decide whether a formula is valid is to enumerate the valuations of its variables. This yields a worst-case exponential time algorithm.

Another approach is to build a proof of the formula to show its validity. This approach is much more powerful, and can be extended to more expressive logics. Among the several formalizations of the concept of a proof and of the automatic derivation of a proof, we will use the sequent calculus described by Gentzen in the 1930's.

2 Sequent Calculus

The sequent calculus is a deduction system to write proofs of formulae. A *sequent* is written $\Gamma \vdash \Delta$, where Γ and Δ are *sets* of formulae. The sequent $\Gamma \vdash \Delta$ means that, under the hypotheses Γ, at least one formula of Δ is valid.

An *inference rule* is made of a set of sequents (called its premises) and a sequent (its conclusion). Graphically, an inference rule is represented by a horizontal line, with the premises above and the conclusion below. Every premise can be the conclusion of another inference rule, and stacking inference rules in this way allows us to build *deduction trees*. The leaves of the trees (graphically at the top) are the premises of the deduction, and its root (graphically at the bottom) is its conclusion. An inference rule with no premises is called an *axiom*, and a deduction with no premises is called a *proof*. So a proof is actually a tree whose nodes are sequents.

To prove that a formula ϕ is valid (i.e. always true), one must find a proof whose root is the sequent $\vdash \phi$, meaning that ϕ is true with no hypothesis.

We use the comma to denote *disjoint union*: Γ, Δ implicitly means that $\Gamma \cap \Delta = \emptyset$, and denotes the union of Γ and Δ. Similarly, when we write A, Γ, we implicitly mean that the formula A is not contained in Γ.

In classical logic, the inference rules are the following:

$$\frac{}{A \vdash A}\,(\mathrm{Id}) \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash A, \Delta}\,(\vdash \mathrm{W}) \qquad \frac{\Gamma \vdash \Delta}{\Gamma, A \vdash \Delta}\,(\mathrm{W} \vdash)$$

$$\frac{\Gamma, A \vdash \Delta}{\Gamma \vdash \neg A, \Delta}\,(\vdash \neg) \qquad \frac{\Gamma \vdash A, \Delta}{\Gamma, \neg A \vdash \Delta}\,(\neg \vdash)$$

$$\frac{\Gamma \vdash A, \Delta \quad \Gamma \vdash B, \Delta}{\Gamma \vdash A \wedge B, \Delta}\,(\vdash \wedge) \qquad \frac{\Gamma, A, B \vdash \Delta}{\Gamma, A \wedge B \vdash \Delta}\,(\wedge \vdash)$$

$$\frac{\Gamma \vdash A, B, \Delta}{\Gamma \vdash A \vee B, \Delta}\,(\vdash \vee) \qquad \frac{\Gamma, A \vdash \Delta \quad \Gamma, B \vdash \Delta}{\Gamma, A \vee B \vdash \Delta}\,(\vee \vdash)$$

Do not be afraid of the formalism. Remember that the sequent $\Gamma \vdash \Delta$ means "given all hypotheses in Γ, we can prove one conclusion of Δ". Imagine that $\vdash$ is like an implication, $\Gamma \vdash \Delta$ meaning $\bigwedge_{\gamma \in \Gamma} \gamma \implies \bigvee_{\delta \in \Delta} \delta$.

3 The project

Implement a simple theorem prover that uses the sequent calculus above.

Your tool should be able to represent formulae, sequents and proofs. Note that a proof is actually a tree, and that a proof can be built recursively.

To help the construction of the proof, note that if the same formula ϕ appears in Γ and in Δ, then $\Gamma \vdash \Delta$ is provable: under the hypothesis ϕ (among others), we can deduce ϕ (among other formulae). This fact can be seen as a new inference rule, called (Id-opt), proven below:

$$\frac{\dfrac{\dfrac{}{A \vdash A}\,(\mathrm{Id})}{\Gamma, A \vdash A}\,(\mathrm{W} \vdash)^*}{\Gamma, A \vdash A, \Delta}\,(\vdash \mathrm{W})^* \qquad \frac{}{\Gamma, A \vdash A, \Delta}\,(\text{Id-opt})$$

Using the rule (Id-opt), you should no longer need the weakening rules $(\mathrm{W} \vdash)$ and $(\vdash \mathrm{W})$.

Note that inference rules "deconstruct" formulae: formulae in the premises are subformulae of the formulae in the conclusion. The idea to build a proof is to deconstruct formulae to obtain the same formula on both sides of the sequent. This procedure terminates: in the worst case, all formulae are deconstructed to atoms (on both sides of the sequent); if the same atom appears on both side of the sequent, then it is provable, otherwise it is not provable.

Let us prove that, if Γ and Δ contain only atoms, and $\Gamma \cap \Delta = \emptyset$, then the sequent $\Gamma \vdash \Delta$ is not provable. Let us assume that it is provable, by contradiction. Since Γ and Δ contain only atoms, the only inference rules that can appear in the proof are the identity (Id) and the weakenings $(\vdash \mathrm{W})$ and $(\mathrm{W} \vdash)$. Consider a branch of the proof: it necessarily ends with a (Id), on some sequent $A \vdash A$, and otherwise contains only weakenings all the way down to

$\Gamma \vdash \Delta$. But since weakenings only add formulae to the sequent, it means that $A \in \Gamma$ and $A \in \Delta$, which contradicts the assumption $\Gamma \cap \Delta = \emptyset$.

During the construction of a proof for a sequent s, if one branch of the proof contains an unprovable sequent containing only atoms, then the proof cannot be completed, meaning that s is not provable.

One simple algorithm to build a proof is to deconstruct all right-hand side formulae, then all left-hand side formulae, until we have only sequents containing only atoms. Feel free to implement it, or to improve it.

We illustrate these principles on the sequent $A \wedge (B \vee C) \vdash (A \wedge B) \vee (A \wedge C)$, shown on Figure 1. Notice that the lower part of the proof contains only right-hand side rules ($\vdash xx$), and the higher part contains only left-hand side rules ($xx \vdash$).

We show a few other (shorter) proofs. First we show the sequent $A \wedge B \vdash A \vee B$:

$$\dfrac{\dfrac{\dfrac{}{A, B \vdash A, B}\,(\text{Id-opt})}{A \wedge B \vdash A, B}\,(\wedge \vdash)}{A \wedge B \vdash A \vee B}\,(\vdash \vee)$$

The proof above first deconstructs the right-hand side of the sequent, then the left-hand side; it is very simple to generate it automatically. Note that other proofs are possible:

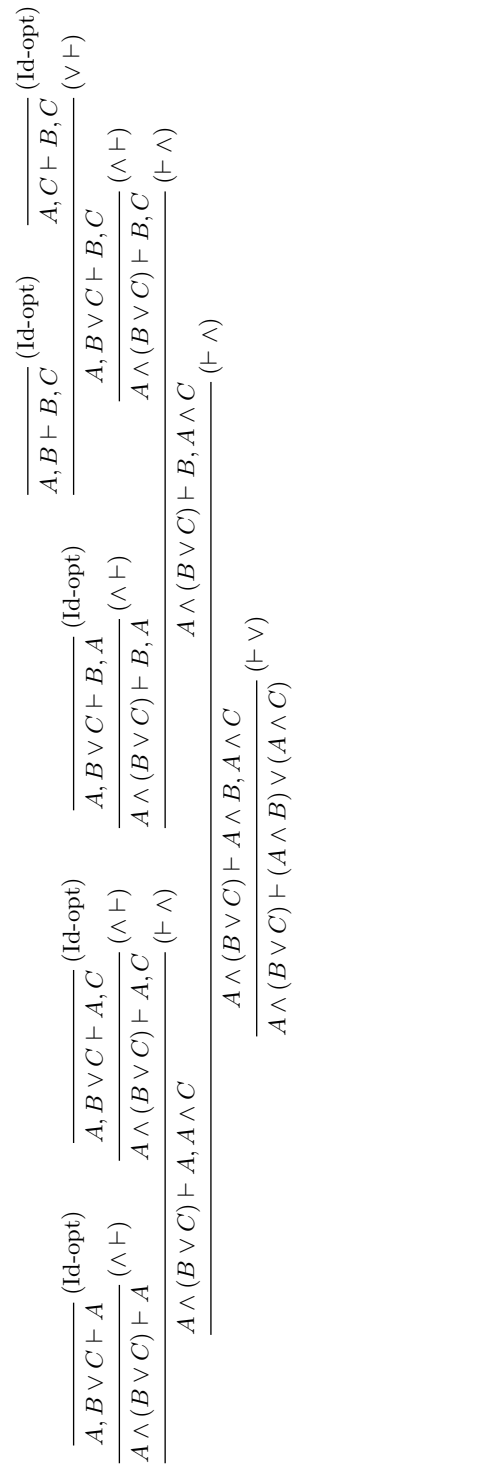
$$\dfrac{\dfrac{\dfrac{}{A, B \vdash A, B}\,(\text{Id-opt})}{A, B \vdash A \vee B}\,(\vdash \vee)}{A \wedge B \vdash A \vee B}\,(\wedge \vdash) \qquad \dfrac{\dfrac{\dfrac{}{A, B \vdash A}\,(\text{Id-opt})}{A \wedge B \vdash A}\,(\wedge \vdash)}{A \wedge B \vdash A, B}\,(\vdash \text{W})}{A \wedge B \vdash A \vee B}\,(\vdash \vee)$$

The two proofs above reorder the inference rules used in the original proof. The weakening ($\vdash$ W) was "hidden" in the (Id-opt) rule of the original proof. Using weakening rules is delicate, as it requires to choose a formula to remove from the sequent. Choosing the wrong formulae can prevent the completion of the proof, and one may think that the sequent is unprovable, where in fact it was the weakening rule who remove the wrong formula. This is why we recommend to rely on the rule (Id-opt) and not use weakenings, as it eases the implementation.

Let us now show that the disjunction is associative (we use binary disjunction and conjunction in this project, but the inference rules could be generalized for n-ary conjunction and disjunction):

$$\dfrac{\dfrac{\dfrac{\dfrac{}{A \vdash A, B, C}\,(\text{Id-opt}) \qquad \dfrac{\dfrac{}{B \vdash A, B, C}\,(\text{Id-opt}) \qquad \dfrac{}{C \vdash A, B, C}\,(\text{Id-opt})}{(B \vee C) \vdash A, B, C}\,(\vee \vdash)}{A \vee (B \vee C) \vdash A, B, C}\,(\vee \vdash)}{A \vee (B \vee C) \vdash A \vee B, C}\,(\vdash \vee)}{A \vee (B \vee C) \vdash (A \vee B) \vee C}\,(\vdash \vee)$$

Figure 1: Proof of sequent $A \wedge (B \vee C) \vdash (A \wedge B) \vee (A \wedge C)$



We now present an incomplete proof of sequent that is not provable.

$$\dfrac{\dfrac{\dfrac{}{A, B \vdash}\,(\text{Id-opt})}{A \wedge B \vdash A}\,(\wedge \vdash) \qquad \dfrac{\dfrac{????}{A, B \vdash C}\,(??)}{A \wedge B \vdash C}\,(\wedge \vdash)}{A \wedge B \vdash A \wedge C}\,(\vdash \wedge)$$

This is the incomplete proof that your tool may build for the sequent $A \wedge B \vdash A \wedge C$. On one branch, we get a sequent containing only atoms: $A, B \vdash C$. For this sequent, there is no rule to apply, so this sequent $A, B \vdash C$ is not provable.

Your tool should also provide a way to parse formulae, using the C syntax for Boolean connectors. A recurrent problem when working with sequent calculus is the difficulty to print proofs. Your tool needs NOT be able to print proofs.

Your tool should be able to take two arguments, representing two formulae ϕ and ψ, and to try to prove the sequent $\phi \vdash \psi$, indicating whether the search for the proof has succeeded or not. For example,

```
scala MyProver 'A & B' 'A | B'
```

should answer `provable`, as the sequent $A \wedge B \vdash A \vee B$ is provable. Similarly,

```
scala MyProver 'A' '!A'
```

should answer `not provable`, as the sequent $A \vdash \neg A$ is not provable.

To test your tool, consider proving classical logical equivalences, such as De Morgan laws, or distributivity laws. To prove the equivalence between two formulas ϕ and ψ, you should prove the sequent $\phi \vdash \psi$ AND the sequent $\psi \vdash \phi$. You can also make sure that your tool detects that $A \vdash B$ and $A \vdash \neg A$ are not provable.