O Uso de Assistente de Provas no Ensino de Lógica

Rodrigo Ribeiro
rodrigo.ribeiro@ufop.edu.br
UFOP

Karina G. Roggia
karina.roggia@udesc.br
UDESC

Cristiano D. Vasconcellos
cristiano.vasconcellos@udesc.br
UDESC

Lógica formal é uma parte essencial do currículo dos cursos de Bacharelado em Ciência da Computação, onde os alunos aprendem a desenvolver deduções matemáticas de maneira precisa. A manipulação de formalismos matemáticos é fundamental para a especificação, projeto e validação de software, como pode ser observado pela presença de componentes eletivos sobre métodos formais nos currículos propostos pela ACM e IEEE [1]. Durante um curso introdutório sobre lógica, espera-se que os alunos resolvam uma grande quantidade de exercícios envolvendo demonstração de teoremas simples envolvendo fatos da lógica de primeira ordem, teoria de conjuntos e indução matemática. Tradicionalmente, os estudantes fazem os exercícios propostos usando papel e caneta. Acreditamos que essa abordagem pode ser aprimorada, uma vez que no modo tradicional, não há um retorno rápido sobre a correção da solução do aluno. É comum que estudantes cometam erros sutis em deduções de forma a invalidá-las por completo. Adicionalmente, é difícil que um professor tenha tempo para identificar e explicar o problema em cada uma das demonstrações escritas por seus alunos. Uma possível solução para esses problemas é o uso de ferramentas computacionais que auxiliem o estudante na construção e verificação de suas deduções. Assistentes de provas são ferramentas de software destinadas a auxiliar na formalização de teorias matemáticas e têm sido usados com sucesso na verificação de software e teoremas matemáticos [2, 3]. Nesse resumo defendemos os benefícios da adoção de assistentes de provas no ensino de lógica formal e como reforço para as habilidades desenvolvidas em fases iniciais de cursos de bacharelado em Ciência da Computação. Como um exemplo de nossa proposta, considere a tarefa de provar a comutatividade da disjunção usando dedução natural para a lógica proposicional:

$$\{A \vee B\} \vdash B \vee A$$

$$\dfrac{\dfrac{}{A \vee B}\ (Id) \qquad \dfrac{\dfrac{\overline{A}\ (Id)}{B \vee A}\ (\vee ID) \qquad \dfrac{\overline{B}\ (Id)}{B \vee A}\ (\vee IE)}{B \vee A}\ (\vee E)}$$

Apesar de ser um fato simples da lógica, esta demonstração possui detalhes que podem induzir o estudante a resolver o exercício de forma incorreta. Para fins de nossa discussão, vamos nos ater em alguns erros cometidos por alunos quando do uso da regra de eliminação da disjunção ($\vee E$):

$$\dfrac{\Gamma, \alpha \vdash \gamma \quad \Gamma, \beta \vdash \gamma \quad \Gamma \vdash \alpha \vee \beta}{\Gamma \vdash \gamma}\ (\vee E)$$

A regra ($\vee E$) permite concluir uma fórmula γ a partir de $\alpha \vee \beta$ se for possível construir duas deduções de γ: uma usando α como hipótese adicional e outra usando β . Alguns descuidos cometidos pelos estudantes em uma demonstração como essa são: 1) não demonstrar a conclusão considerando como hipótese adicional os componentes esquerdo e direito da disjunção; 2) incluir ambos os argumentos da disjunção como hipóteses em ambas as deduções da conclusão; 3) deduzir a conclusão apenas uma vez e não duas como especificado na regra ($\vee E$). A seguir apresentamos o código Coq equivalente à dedução anterior.

```
Theorem or_comm {A B : Prop}(H : A \/ B) : B \/ A.
Proof.
   destruct H as [HA | HB]. (** regra \/E *)
   - right.      (** regra \/ID *)
     assumption. (** regra Id *)
   - left.       (** regra \/IE *)
     assumption. (** regra Id *)
Qed.
```

O uso de um assistente de provas como Coq para construir deduções pode auxiliar o estudante na solução do exercício anterior por eliminar todos os "descuidos"mencionados anteriormente. Em especial, o assistente de provas evita a construção de deduções inválidas por considerá-las erros de compilação quando da verificação do código. Além disso, o uso de um assistente de provas permite a construção interativa da demonstração, o que permite ao aluno compreender os passos dedutivos utilizados para obter a conclusão desejada.

AGRADECIMENTO

O presente trabalho foi parcialmente financiado pela Fundação de Amparo à Pesquisa e Inovação do Estado de Santa Catarina (FAPESC).

REFERÊNCIAS

[1] Alison Clear, Tony Clear, Abhijat Vichare, Thea Charles, Stephen Frezza, Mirela Gutica, Barry Lunt, Francesco Maiorana, Arnold Pears, Francois Pitt, Charles Riedesel, and Justyna Szynkiewicz. 2020. Designing Computer Science Competency Statements: A Process and Curriculum Model for the 21st Century. In *Proceedings of the Working Group Reports on Innovation and Technology in Computer Science Education* (Trondheim, Norway) *(ITiCSE-WGR '20)*. Association for Computing Machinery, New York, NY, USA, 211–246. https://doi.org/10.1145/3437800.3439208

[2] Georges Gonthier, Andrea Asperti, Jeremy Avigad, Yves Bertot, Cyril Cohen, François Garillot, Stéphane Le Roux, Assia Mahboubi, Russell O'Connor, Sidi Ould Biha, Ioana Pasca, Laurence Rideau, Alexey Solovyev, Enrico Tassi, and Laurent Théry. 2013. A Machine-Checked Proof of the Odd Order Theorem. In *Interactive Theorem Proving - 4th International Conference, ITP 2013, Rennes, France, July 22-26, 2013. Proceedings (Lecture Notes in Computer Science, Vol. 7998)*, Sandrine Blazy, Christine Paulin-Mohring, and David Pichardie (Eds.). Springer, 163–179. https://doi.org/10.1007/978-3-642-39634-2_14

[3] Xavier Leroy. 2009. Formal verification of a realistic compiler. *Commun. ACM* 52, 7 (2009), 107–115. https://doi.org/10.1145/1538788.1538814

EduComp '21, Abril 26–30, 2021, Jataí, Goiás, Brasil (On line)